AD-A115 697   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC   F/G 15/7
THEATER WARFARE PROGRAMS AT AFIT: AN INSTRUCTIONAL AID.(U)
MAR 82   A WAISANEN
UNCLASSIFIED   AFIT/GST/OS/82M-15   NL

1 OF 2
AD A115697

1

THEATER WARFARE PROGRAMS AT AFIT:
AN INSTRUCTIONAL AID

THESIS

AFIT/GST/OS/82M-15

Anthony Waisanen
Captain USAF

Approved for public release; distribution unlimited.

AFIT/GST/OS/82M-15

THEATER WARFARE PROGRAMS AT AFIT:
AN INSTRUCTIONAL AID

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Anthony Waisanen
Captain USAF
Graduate Strategic and Tactical Sciences

March 1982

Approved for public release; distribution unlimited.

Preface

One of the courses offered by the Air Force Institute of Technology (AFIT) is the Combined Warfare course (ST7.01) which is designed to teach concepts of warfare. The first year this course was offered, the class was sent to Maxwell AFB to participate in the playing of the Theater Warfare Exercise (TWX). This proved to be an invaluable addition to the course. Unfortunately, the exercise was designed solely for operation on a Honeywell computer. If the exercise were to run at AFIT, it would need to be compatible with at least one of the computers at Wright-Patterson AFB; the CDC or the Harris.

This report describes the modifications which were required to make the programs, data and operation compatible with the CDC computer.

I wish to thank Col Don Stevens for suggesting this topic. Hopefully, future students will derive as much insight from TWX as I have. I am indebted to Maj Dan Fox for his patience and suggestions during the many hours we spent discussing this problem. I could not have completed half of this thesis effort had he not introduced me to the UEDIT text editor and the PDP-11/60.

I also wish to thank the computer personnel at HQ AU/ACDY and AFIT/AD for taking the time to talk about how the TWX programs were designed and how they operate. In particular, I wish to thank Lt Ed Laugel for the many hours he spent explaning the origins of the TWX and the operation of Honeywell computers.

Capt Carl Lizza is also to be thanked for his assistance in the segmentation process and debugging of the programs. SrA John D. Long's help and knowledge of the CDC was greatly appreciated. AFIT would not have the TWX if it were not for these people.

I wish to thank my fellow classmates who suggested modifications, spent considerable effort in developing a test data base, and helped in so many other ways.

Finally, my wife Holly is to be thanked for her patience and understanding, and support through the many difficult periods during this project.

Anthony Waisanen

TABLE OF CONTENTS

List of Abbreviations

Abbreviation	Definition
AAFCE	Allied Air Forces, Central Europe
AFIT	Air Force Institute of Technology
AG	Air to Ground interface program
APE	AAFCE Planning Executive program
AR	Air battle simulation program
CAWC	Combined Air Warfare Course
CDC	Control Data Corporation (computer)
IP	Input Print program
LA	Logistics Analysis program
LB	Land battle simulation program
LI	Land order Input program
MA	Merge Actions program
MI	Mission order Input program
MR	Merge Records program (2ATAF and 4ATAF)
OR	Overrun program
SQ	Merge Records program (Blue and Red sides)
TWX	Theater Warfare Exercise
XX	Original main driver routine

List of Tables

Abstract

This thesis report details the processes and modifications that were required in order to execute the Theater Warfare Exercise (TWX) on the CDC computers at Wright-Patterson AFB. Prior to this effort, the TWX programs and data files could only be accessed and executed with Honeywell computers.

By modifying the data files, program coding, overlaying techniques, and operation, the TWX can now be run on any CDC computer. Any other computer with sufficient central memory and an ANSI standard FORTRAN-77 compiler can also execute the programs provided the operation methods (procedure files) are modified.

THEATER WARFARE PROGRAMS AT AFIT:
AN INSTRUCTIONAL AID

I Introduction

One of the objectives of a military institution like the Air Force Institute of Technology (AFIT) is to give its graduates the education necessary for them to be able "to understand their cultural and technological environment and to analyze and attempt to solve its problems (Ref 1:2+)." Ideally, the students should have the opportunity to apply what they have learned prior to graduating. Thus, their analyses could be critqued by other analysts in a controlled setting. Such an interaction could instill in the student an understanding of the role and process of analyses in the military. Students in the School of Engineering have very full class schedules. Since non-computerized exercise would take too much time and effort to be useful, some computerization is needed. Thus, attention is directed to computerized exercises.

Background

One way of gaining this experience is to participate in the Theater Warfare Exercise (TWX) which is played as part of the Combined Air Warfare Course (CAWC) at Maxwell AFB. This is not a practical solution however considering the time-intensive class-work required for all AFIT courses. Nor, due to TDY expenses, is it financially feasible since the TWX typically requires at least two days for introduction and preplanning and five days of execution for a minimum of seven days. This was done in February 1981 with the first class of Combined Air Warfare Seminar (ST7.01) students but there were only four students taking the course. Future classes are expected to number in excess of 15.

Another alternative would be to use a simulation in existence at AFIT such as STAG (Ref 4). However, STAG is only a two-player simulation with very limited realism and interaction. It was not designed to represent real-world results (Ref 4:13).

The most desirable option would be to perform the TWX at AFIT. This option offers a realistic exercise without excessive disruption to the schedules of the players and without requiring a large TDY budget. This would require the modification of the programs currently being used for the CAWC to a set of programs which could be run on either the Harris or CDC computers at Wright-Patterson AFB.

Problem Statement

Prior to this effort, the TWX programs could only be run on Honeywell computers. If these programs are to be used at AFIT, they must be able to execute on a variety of computers; specifically, the Harris and CDC computers. This required that the coding and data be transferred to one of these computers.

It was decided to convert the programs to ANSI standard

FORTRAN-77. This resulted in enhanced portability. FORTRAN-77 was selected since all of the TWX programs use character variables and ENCODE and DECODE statements (which are the FORTRAN-77 equivalent of internal file WRITEs and READs).

A significant effort was required to develop job control language (JCL) to access the data files and execute the programs. In the original programs (Ref 7), data file access was accomplished with Honeywell machine dependent code in the programs and not with the JCL. Since file manipulation is strictly a function of the type of computer system being used, this was changed so that file manipulation would be completely external to the programs. Thus, transportation of the TWX programs from one computer system to another could be accomplished without requiring modification of the TWX source code.

Scope and Limitations

The purpose of this thesis effort was to create a set of theater level warfare exercise programs at AFIT that are functionally identical to those used in the CAWC at Maxwell AFB on August 1981. That is, from a player's and analyst's point of view the programs should appear to operate in identical fashion to those used in the CAWC. This way, players, analysts, and exercise administrators who are familiar with one set of the TWX will be able to function with the other set without lengthy learning periods. This also permits the use at AFIT of program documentation and exercise directions which have already been dev3loped for the exercise at Maxwell AFB. This effort was limited to the transfer and modification of the data files and program coding required to execute the programs at AFIT. None of the algorithms used in determining troop movement, losses, and others were modified unless the program would not otherwise run.

Because of its accessibility and large storage capacity, the CDC computer was used in making the modifications and developing the JCL.

Documentation of this thesis effort is was limited to the description of that coding which had to be modified. According to one of the TWX programmers (Ref 6), the original directives to the TWX programmers included the requirement for extensive documentation of every routine. Therefore, this documentation together with the documentation contained in this thesis should adequately describe the modified programs.

Finally, modifications to the coding were as limited as possible and no modifications were made to the data file formats. When system-specific intrinsic routines were found (for example, routines which determine binary file size), equivalent FORTRAN-77 coding was developed. The resulting code was placed in a seperate library, not in the TWX programs themselves. This way, the translated TWX programs appear to be as similar as possible to the original and all system-specific coding is located only in the library. In those instances where a routine could not be recoded because of name similarity (for example, ENCODE), the program was modified by changing the offending code into comment statements and putting the equivalent coding directly after it.

Calls to some routines like FMEDIA and CREATE are required only by the Honeywell computers (Ref 5). These calls were changed to comment statements.

Along with the TWX programs, source code for 25 utility programs were also transfered. These programs are used at Maxwell AFB to manipulate or modify the data bases. Because they are highly system-specific and contain coding which performs JCL functions, they have not been converted and, in some cases, similar programs have been developed. Their code has been retained on file for future reference.

II Modifications

There were three major areas involved with modification; data transfer, coding modification, and verification of the modified coding. Data transfer involved the conversion of the extensive data bases used by the TWX programs from a Honeywell-specific format and data representation to a CDC-specific format and data representation. Coding modification involved the identification of Honeywell-specific coding and the development of equivalent code which is not specific to one computer. Verification was limited in that only those routines which were developed to replace the Honeywell intrinsic routines were thoroughly tested; the major TWX programs were verified to the extent that their operation was identical to the originals.

Data Transfer

Two types of data files are used in the TWX programs; sequential card image and random access binary (Ref 2). The transfer of the card image files was only a matter of reading the magnetic tape; the binary files required considerably more modification. Since the original data was stored as 36-bit/word binary, it had to be read in binary form and then written to tape as card image. The program used to do this read and wrote only real values. Thus, character variables such as the target names in file RLUM had to be input manually. Other than this problem, the conversion into 60-bit/word binary form only required short programs that read card image and wrote binary. A different program was written for each format of data file. Each of these transfers was verified using a similar program that read binary and wrote card image. The output from these programs was compared to the card image listings obtained during the file transfers.

Program Modification

There are three types of programs in TWX; batch, interactive, and utility. These programs and the library routines used to replace Honeywell specific system routine calls differ greatly in both function and form and so will be discussed seperately. First, however, is a general discussion of the major compatibilities and incompatibilities between the Honeywell FORTRAN compiler and the ANSI standard FORTRAN-77 compiler.

General. There were two types of modifications to the TWX programs; modifications which were compatible with both the Honeywell and CDC computers and modifications which were incompatible with the Honeywell FORTRAN compiler.

Compatible Modifications. One of the differences in compilers is the maximum allowable size of labels and variable names. In Honeywell FORTRAN, a maximum of 8 characters is allowed for variable, entry, subroutine, and program names. To minimize the dissimilarity between the original and modified versions, the seventh and eighth characters were deleted since only 6 characters are allowed by an ANSI standard FORTRAN-77 compiler. Only when a conflict would result were any of the other characters altered.

Another difference is that the Honeywell compiler allows character and numeric variables to be in the same COMMON block. This is not allowed in FORTRAN-77. When this occurred, the character variables were placed in a seperate but similarly labelled COMMON block (Table I).

TABLE I

COMMON Block Modification

	Original	Modified	Commons
Program	Common	Numeric	Character
AG	ACCNTR	ACCNTR	ACCNTC
APE	LAINB	LAINB	LAINBC
AR	BUFFER	BUFFER	BUFFEC
	CNTROL2	CNTRL2	CNTR2C
LA	ACDATA	ACDATA	ACDATC
	LGDATA	LGDATA	LGDATC
LB	PAGEDATA	PAGEDA	PAGECH
	TABLDATA	TABLDA	TABLCH
MI	CONTROL	CONTRO	CONTRC
	CYCLE	CYCLE	CYCLC
	READIN	READIN	READIC
OR	OUTPUT	OTPUT	OTPUTC
	TGOUT	TGOUT	TGOUTC

In some cases source code records extended past the 72nd column after being formatted for the FORTRAN-77 compiler. For example, a record is identified as a continuation by the Honeywell compiler if an ampersand is in the first through sixth column. The FORTRAN-77 compiler identifies a continuation record by the presence of any character in the sixth column. This meant that all continuation lines that did not begin in the sixth column had to be reformatted to be compatible with the FORTRAN-77 compiler which made some records more than 72 characters long, the maximum record length. In these cases, the extended coding was continued on the following line.

Probably the most troublesome difference between the computer systems is that the Honeywell loader initializes all variables to 0. Since the CDC initializes all variables to negative indefinite, all variables in labelled commons were set to 0 unless they were defined by the program.

Incompatible Modifications. Not all of the modifications made to the TTWX programs at AFIT are compatible with the Honeywell FORTRAN compiler. For example, free-field READ and PRINT statements in Honeywell FORTRAN are:

```
READ ,xxxx
PRINT ,xxxx
```

but in ANSI standard FORTRAN-77 they are:

```
READ *,xxxx
PRINT *,xxxx
```

Similarly, direct access files are read and written in Honeywell FORTRAN using the statements:

```
READ (lud´rec) xxx
WRITE (lud´rec) xxxx
```

where

lud = device number

rec = record number

The FORTRAN 77 compiler, however, uses the statements:

```
READ (lud,REC=rec) xxxx
WRITE (lud,REC=rec) xxxx
```

These statements occur infrequently only in the interactive programs so modification of the coding back to a form compatible with the H6000 compiler is not difficult.

The original programs made extensive use of intrinsic routines which would modify the bits of certain variables. Since this type of operation depends directly on the number of bits per word being used by a given computer, the routines were recoded and placed in the library file. Instead of modifying the bits of a numeric variable, these new library routines modify the characters of character-type variables (with each variable consisting of 36 characters which represent the original 36 bits/word). This implimentation allows the modified programs to run on any machine using ANSI standard FORTRAN-77 regradless of word length. This means that the modified use of Boolean variables and Honeywell intrinsic routines is not easy to modify back to the original.

All calls to the Honeywell system routines "LINK" and "LLINK" have been changed to comment statements since they operated the Honeywell overlay procedures.

Finally, every effort was made to exclude system-dependent coding and functions. However, some routines cannot realistically function without them. For example, subroutine DATIM, which returns the current date and time, uses two CDC intrinsic functions "DATE" and "CLOCK." These may require recoding if another system is to be used.

Batch Programs. The purpose of the batch programs is to perform the simulation of a theater-level war using the player-defined inputs. The players only see the results of these programs which they use in planning the following day´s missions. Since these programs are never directly accessed by the players, data file access is performed by JCL rather than by program coding and no input/output routines required coding modification. One modification which was required was the use of logical variables.

One of the options indirectly available to the players is the number of reports that will be printed by the programs AR, LB, OR, and LA (words 1 through 19 on records 2 and 3 of file RMRx# control this). The option is indirect in that the values can be changed but a utility program must be used. These values are used as counters and sometimes as logicals with integer 0 being interpreted as "FALSE" by Honeywell FORTRAN. Since the CDC FORTRAN-77 compiler interprets all non-negative values to be "FALSE", the logical variables (words 1, 11, 12, 13 and 15) have been set to -1. Since words 8 and 9 are used by LB as logicals and integers, they have also been set to -1 and the coding of subroutines REP08 and REP09 have been modified to use their absolute values. The resulting coding will thus function on the CDC and Honeywell computers.

The air battle simulation program AR required more extensive modifications. Originally, it was designed to simulate tactical airlift (TAL), air force augmentation (AUG) and aircraft role change (RC) functions as well as air battle simulation. Since the TAL, AUG, and REC functions are now performed by APE, the subroutines DITAL, DUAUG, and DIRC are no longer needed according to CAWC personnel at Maxwell AFB (Ref 9). They have been commented out.

Finally, the use of maximums was modified in AR. In TWX, there is a maximum number of bases/side, aircraft types/side, cycles/day, munition types/side, corps/side, days/exercise, and seminars (words 13 through 20 of record 4 of RMRx#). First, the array size of variable MAX was reduced from 9 to 8 since the FORTRAN-77 compiler does not permit the reading of more words per record than were specified when the file was declared. Second, the actual number of bases/side, aircraft types/side, and so forth were used as counters in place of the equivalent values of MAX.

Interactive Programs. Some of the important features of TWX are the allocation and apportionment of air resources, movement of materiel, and mission definition. These functions are performed by three interactive programs; APE (AAFCE Planning Executive program), MI (Mission Input program), and LI (Land Input program). All of these programs prompt the player for directions and modify the appropriate data files; hence, the term, interactive. These programs all had subroutines which determined the files to be accessed based upon the side, seminar number, and, for mission inputs, the ATAF number. After this information was input, the files were accessed using system dependent code within the programs. Since this function is performed with JCL on the CDC, these subroutines (Table II) were removed from both programs and the program INPTR was developed (Appendix A).

TABLE II

Deleted TWX JCL-type Subroutines

Program	Subroutines
APE	PIATCH, PILO, PILOPS, PILOSEM and PILOSIDE
LI	PILOSEM and PILOSIDE
MI	PIATCH, PILO, PILOPS, PILOSEM and PILOSIDE

It should be noted that INPTR, together with the JCL necessary to compile it and catalog the object deck, comprise the file INPTGO in the Indirect File System (IFS) file TWXRUN.

The purpose of INPTR is to interactively construct the procedure file XECUTE. If the player correctly inputs the information together with the appropriate password, XECUTE will execute either of three procedure files APEGO, MIGO, or LIGO (these are discussed in Chapter IV). Thus, converting the programs to operate on a different computer will only involve modifying these three procedure files and not any of the programs (assuming the other computer has a FORTRAN-77 compiler). If the player fails to input the correct password in three tries, XECUTE will copy a message for the player to get assistance and then terminate.

When a file is written or modified using a Honeywell computer, the new information is made permanent when information is read from the same area of the file. For this reason, the dummy array FLUSH is often read after a number of "writes". According to the programmers, this would allow continuation in the event of accidental disruption without requiring a complete restart (Ref 6). This activity was replaced by the use of the JCL statements REPLACE and EXIT,S. If the program (APE, MI, or LI) runs successfully, the written file is put in an indirect access file. If the program is not successful and an abort condition is detected, the skipping of JCL is stopped by the EXIT,S statement and the same files are made permanent. This allows the player to continue from the point immediately prior to the error rather than requiring a total restart.

Utility Programs. Since 16 out of the 24 data files used by TWX are binary direct access, modifications to them can only be made by using interactive programs. Rather than have one program capable of using a variety of formats, a different program was written for each format. This resulted in ten read/read and modify pro ams (Table III). These programs are all capable of reading their applicable file(s) and many offer modification options. They all assume that the file to be read/modified is accessible as device 1 (in other words, "TAPE1").

TABLE III

Read and Modification Utility Programs

Program	Applicable File
RABREAD	RABx#
RACREAD	RACx
RAPREAD	RAPx#, RPPx# (no changes permitted)
RBLREAD	RBLx#, RRLx#, RMUx
RCRREAD	RCRx#
RLUREAD	RLUx#
RMRREAD	RMRx# (no changes permitted)
RPPREAD	RPPx#
RTGREAD	RTGx#
R2MREAD	R2Mx#, R4Mx#, RLGx and RMRx# (logicals are displayed as integers)

Using these programs to modify a file requires the use of the TWX Data Base Manual (Ref 2) since these programs print only values of the data, not meanings.

The operation of these utility programs is not automatic. Unlike the simulation programs which have procedure files to access the necessary tapes, the utility programs must be manually accessed, compiled and executed by the user with its appropriate input file also manually accessed by the user. Thus, deliberate and purposeful activities preceed the modification of ıata files. This is to prevent accidental modification to the data files. All utility programs are stored in the IFS file TWXUTIL.

Library Routines

One of the goals of this effort was to minimize the modifications to the program code. Since all of the programs at some time or other use routines which are intrinsic only to a Honeywell computer, this meant that equivalent coding for those routines had to be developed and added to a TWX library called TWXLIB (Appendix B). Thus, instead of modifying the program coding, this library containing routines with the same names as Honeywell specific system routines is declared and the equivalent functions are performed.

Verification

There are two types of coding now in the TWX programs; modified coding and completely new routines (library routines) which were developed at AFIT. Since no documentation was available for the original set of programs, verification of the modified coding was limited to verifying that the programs executed like the originals. The library routines were all verified using test programs which would execute each aspect of each routine. Since these programs were only for testing purposes, they were retained until the verification was completed and, thus, are no longer on file.

III Segmentation

The major interactive programs APE, MI, and LI have two features in common: 1) a central memory requirement in excess of 65K (the maximum interactive core limit for the CDC) and 2) they were originally written in a highly structured form with each subroutine either modifying a particular set of arrays or checking and editing input data. This implied that these programs could be run where only the main driver and a very few subroutines would be in central memory at any one time without unduly increasing the run time. This, in turn, implied that some method of overlaying could be used to decrease the central memory requirement. Of all possible methods of overlaying, segmentation was chosen because of its transportability, greater flexibility (Ref 3:7-1), and because it does not require changes to the source programs.

The specific methods of segmentation are described in Chapter 7 of the CYBER LOADER VERSION 1 Reference Manual (Ref 3) but are not easily understood. Along with the specific rules for segmentation, the following rules have been found to be helpful:

1. In a TREE directive, the label field must not contain a program, subroutine, entry point, or function name that is in the object deck to be segment loaded. Also, this name must appear in the specification filed of another TREE directive. If it does not, it will be interpreted as another root segment.
2. A function or subroutine which is to be used by more

than one segment must be in the directive field or fields of one or more INCLUDE directives; it cannot be in both an INCLUDE directive field and a TREE directive field.

3. The label field of an INCLUDE field must be a function, subroutine, or program name and cannot be an arbitrary label or entry point name.

The first step taken in segmenting the TWX programs was to determine the subroutine interactions (that is, which subroutines call which other subroutines) and then construct a tree depicting those interactions.

The major tree (or "root segment") was constructed by using the main program as the root and the subroutines called by it as the branches. In the case where a subroutine in a branch would call other subroutines, another tree was made using this subroutine as its root and the subroutine name in the original tree was replaced with the label of this new tree. This procedure was used in segmenting APE, MI, and LI.

For example, the AAFCE Planning Executive Program, APE, calls subroutine PI (file initialization), OPTN (file modification option processor), and RAPUP (final data base modifications). The first two subroutines both call numerous subroutines and operate independently so both were declared as trees. This tree structure was converted into a set of SEGLOAD directives for processing by the CDC loader. In order to segment a program, the source is first compiled, the object deck is loaded and then operated on by the SEGLOAD directives. The segmented object deck is then cataloged.

To automate this process, procedure files of JCL and SEGLOAD directives were developed to convert the source decks of APE, MI and LI into segmented object decks (Appendix C). The procedure files also catalog the resulting decks.

To execute any one of these procedure files, the command is

```
BEGIN,pname,lfn1,lfn2.
```

where:

pname = procedure name (APESEG, MISEG or LISEG)

lfn1 = local file name of the procedure file

lfn2 = local file name of the source deck (APE, MI, or LI)

If the local file name is the same as the procedure name, the procedure file is executed with the command:

```
pname,lfn2.
```

where:

pname = procedure name (APESEG, MISEG or LISEG)

lfn2 = local file name of the source deck (APE, MI, or LI)

IV Operation of the Modified TWX Programs

There are two groups of people who need to be involved whenever the TWX is executed: players and administrators. The players are those people who perform the analyses of the outputs, develop the air and land orders used in the simulation and input the data by executing the programs APE, MI, and LI. The activities of the players is described in the Theater Warfare Exercise (TWX) Players´ Handbook (Draft Copy) (Ref 8).

The administrators are responsible for the initialization of the exercise and the direction of student activities. They also represent the "higher authority" capable of authorizing increases of forces and use of nuclear resources. These are the people responsible for the operation of the TWX, execution of the batch programs SQ, MR, IP, AR, AG, MA, LB, OR, and LA, and resetting of the file lock flags and exercise day indicators (Table IV). They are also responsible for the execution of the procedure SETUP (Appendix D) and the maintenance of the TWX files (Appendix E).

TABLE IV

Daily File Modifications

File	Variable Location		Reset to
	Record(s)	Word(s)	
RAPW#	5 - 48	1	-1 ("TRUE")
RBLW#	1	9	0.0
RRLW#	1	9	0.0
RLGW	2 - 3 2 - 3	1 - 2 16	Current Day 0
RLUW#	1	6 - 7	Current Day
RMRR#	1	7	Current Day
R2MW#	2 - 3 2 - 3 4 - 265	1 - 2 3-5,13,16-17 1 - 20	Current Day 0 0
R4MW#	2 - 3 2 - 3 4 - 265	1 - 2 3-5,13,16-17 1 - 20	Current Day 0 0

General

The operation of the TWX is in two phases: initialization and execution. Initialization is essentially the set-up of the necessary files and object decks. It also includes the running of programs to produce the data needed by the players for planning. The execution phase invloves the cyclical execution of interactive and batch programs to simulate the daily publishing and execution of air and land orders. The initialization phase occurs any time prior to the exercise while the execution phase lasts for a maximum of five simulated days.

Initialization

The first step in executing the TWX is the construction of all required object decks and libraries (Table V). This can be performed manually but the procedure file SETUP (Appendix D) has been developed for this purpose. To execute this procedure file, it is necessary to make a local copy using the IFS command "GET,SETUP,ID=TWXRUN" and then execute it with the statement "SETUP."

TABLE V

TWX Object Decks and Libraries

Programs	Object Decks
APE	APELGO
AR	ARLGO
LB	LBGO
LI	LIGO
MI	MIGO
Side	**Libraries**
BLUE	DATA11, DATA12, DATA13
RED	DATA21, DATA22, DATA23
Both	BACKUP, BATCHIN, MASTER

The next step is to execute the Weapon System Summary program (WSS). This too has been automated; the file WSSGO is retrieved from the indirect library file TWXRUN and then executed with the statement "WSSGO." The output from this program is stored in the indirect library file PRINT and is needed by the players for the duration of the exercise (it contains such information as the aircraft names, weapon loadings, and effectiveness indices).

After the initial introduction to the exercise, the players enter the preplanning stage. This stage requires that the batch programs AR, AG, MA, LB, OR, and LA be sequentially executed. The output files ARPLB1, ARPLR1, LBPLB1, LBPLR1, ORP2B1, ORP4B1, ORP2R1, ORP4R1, LAP2B1, LAP4B1, LAP2R1, and LAP4R1 are then retrieved from the indirect library file PRINT and given to the players for their preplanning. The execution phase of the exercise now begins.

Execution

The execution phase of the TWX begins with the players performing their mission planning and entering data with the interactive programs APE, MI and LI. When these programs have been executed, the batch programs SQ, MR, IP, AR, AG, MA, LB, OR, and LA are sequentially executed and the output files are given to the players for the next day's mission planning (each simulat on day consists of a day and a night cycle). This phase continues until the exercise administrators terminate the exercise for a maximum of five simulation days.

In the original TWX, MR through LA were all major subroutines of a single program, XX (the source for the main driver XX is retained as TWXSRCE in the indirect library file TWXPROG). These programs are run seperatedly to improve flexibility and to decrease turn-around time. In fact, only AR and LB require too much central memory to be executed in time sharing mode (it should be noted that OR does require segmentation to execute in time sharing mode). Rather than have an executive program which sequentially executes the batch routines, the JCL for the modified programs, beginning with AR, accesses the procedure file for the succeeding routine and executes it. These procedure files create and batch the JCL necessary to execute the succeeding program. Each of these procedure files has the suffix GOB and is stored in the indirect library file TWXRUN. SQ, MR, and IP, however, must be executed individually with procedure files SQGO, MRGO, and IPGO respectively.

After all batch programs have been executed, the appropriate data files are replaced and the cycle begins again with the players performing analysis and planning and again entering data using the interactive programs APE, MI and LI.

V RECOMMENDATIONS

Because of the amount of time which was required to translate the data and programs into a modifiable form, some problem areas could not be addressed with this thesis effort. This includes areas such as validation of the original algorithms, streamlining of the batch programs, modification of the land battle program, and complete and thorough documentation of all programs.

The validation and documentation of the original coding was not performed because the necessary analyst's manuals are either non-existent or are too scarce to be made available. The set of TWX programs used in the CAWC are continually being modified. Thus, it is recommended that the programs at AFIT be documented without referrence to future manuals from Maxwell AFB.

The air and land battle programs (AR and LB respectively) both have central memory requirements in excess of 225K. Because of this and the large amounts of central processor and input/output time required, both jobs have very long turn-around times. One was of streamlining both programs would be to segment them. There are two procedure files in the indirect library file TWXRUN which have been designed to compile, segment, and catalog the segmented object decks of AR and LB. These procedure files are ARSEG and LBSEG respectively. Time did not permit the final verification of either of these two procedure files.

Currently, the land battle program and data files are not designed to provide information regarding future strategies.

This requires the players who are planning ground-support air sorties to guess what their ground forces will be doing during the next day of simulation. This clearly needs to be modified. A suggestion would be to modify the printout so it provides not only the location of the forces, but also its direction.

There is no automatic resetting of data files. Whenever APE, MI or LI is executed, some data file values are changed to indicate the status of the user. If the program terminates normally, these values (lock flags) are set to 1 indicating that the user had completed the desired modifications of those data bases. If those data bases are used by the same program, the user will be denied access to them until the lock flags are set to 0. Before another day of simulation can begin, the lock flags in files RBLW#, RRLW#, RLGW#, R2MW#, and R4MW# must be set to 0 and the exercise day updated in files RLGW#, RMRR#, R2MW#, and R4MW#. The program II (Input Initialization) was designed to perform this operation on the Honeywell, but II was not modified because of its highly specific coding. Currently, these file modifications are made manually using the utility programs described in Chapter II.

Finally, there is one unresolved problem. When either a segmented or unsegmented version of MI is executed and an invalid offensive counter air (OCA), battle area interdiction (BAI), or interdiction (IND) order is input, an error condition is detected and the player is asked if the order is to be changed. Regardless of whether the answer is "Y" or "N", a Mode 1 error occurs (an illegal address is specified). This condition appears to be the result of the AO register for subroutine EDITC1 being modified by subroutine CYFMT1. The addition of "PRINT" and "CONTINUE" statements has not changed the error condition; the error remains in the address for the variable, INTSO2.

Bibliography

1. Air University. *AFIT 1979-81 Catalogue*, *18*: (undated).

2. Air University. *TWX Data Base Manual*. Maxwell AFB: Alabama, undated.

3. Control Data Corporation. *CYBER Loader Version 1 Reference Manual* (Revision G). California: Control Data Corporation (1979).

4. Foley, John M. *STAG: A Two Person Simulated Tactical Air War Game*. Air Force Institute of Technology: Wright-Patterson AFB, Ohio, March 1980. (AFIT/GST/OS/80M-2).

5. Honeywell. *Fortran*. Computer manual. Minneapolis: Honeywell Corporation, January 1975.

6. Laugel, Edmund L. Conversations. Wright-Patterson AFB: Ohio (9 July 1981 to 9 March 1982).

7. Ritchey, Conrad. *et*. *al*. TWX computer program listings. Maxwell AFB, Alabama (12 August 1981).

8. Waisanen, Anthony. *Theater Warfare Exercise* (*TWX*) *Players´ Handbook* (Draft Copy). School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1982.

9. White, Fred. Telephone conversation. Maxwell AFB: Alabama (9 March 1982).

Appendix A

INPTGO Procedure File/Program Listing

```
.PROC,INPTGO.
.*  THE PURPOSE OF THIS PROCEDURE FILE
.* IS TO COMPILE THE PROGRAM "INPTR" AND
.* THEN CATALOGUE THE OBJECT DECK AS "INPTRLGO"
.* UNDER THE USER´S ACCOUNT NUMBER.
.*
.*  THE FIRST STEP IS TO RETURN ANY LOCAL FILE
.* NAMED "LGO."
RETURN,LGO.
.*  THEN "LGO" IS REQUESTED.
REQUEST,LGO,*PF.
.*  INPTR IS NOW COMPILED.
FTN5,I=PROG,LO=0,DB.
.*  THEN THE OBJECT FILE IS CATALOGUED
CATALOG,LGO,INPTRLGO,XR=FOX,PW=FOX,RP=999.
.*  THE LOCAL FILE AREA IS CLEANED.
RETURN,LGO,PROG.
.*  CONTROL IS RETURNED TO THE USER.
REVERT.
.*
.*  THIS LINE DEFINES LOCAL FILE "PROG"
.* WHICH WILL CONTAIN THE SOURCE OF INPTR
.DATA,PROG.
```

```
      PROGRAM INPTR
      COMMON/FILEDA/ICHOIS, ISEM, ISIDE, IATAF
      COMMON/ITIMS/ITRY
      DATA ICHOIS/0/, ISEM/0/, ISIDE/0/, IATAF/0/
      OPEN(1,FILE='XECUTE',ACCESS='SEQUENTIAL')
      REWIND 1
CX  FIND OUT WHICH PROGRAM IS TO BE RUN
 50   PRINT 12
      READ *,ICHOIS
      IF( ICHOIS .LT. 4 ) THEN
        IF( ICHOIS .LT. 1 ) GO TO 50
            ITRY = 0
            CALL INTER
        IF( ITRY .LT. 3 ) THEN
          CALL BUILD
          CALL INFORM
        ELSE
          CALL DONE
        ENDIF
      ELSE
          CALL STOPR
      ENDIF
      STOP
 12   FORMAT(' DO YOU WANT TO RUN THE:',/,
     &'   1) AAFCE PLANNING PROGRAM',/,
     &'   2) MISSION PLANNING PROGRAM',/,
     &'   3) LAND MOVEMENT PROGRAM',/,
     &'   4) STOP    >')
      END
      SUBROUTINE BUILD
      COMMON/FILEDA/ICHOIS, ISEM, ISIDE, IATAF
CX  THIS SUBROUTINE BUILDS THE APPROPRIATE JCL FILE
CX   TO ACCESS ALL NECESSARY DATA FILES.
CX BRANCH ON TYPE OF PROGRAM
      IF( ICHOIS .EQ. 1 ) THEN
        WRITE(1,11) ISIDE, ISEM, IATAF
      ELSE IF( ICHOIS .EQ. 2 ) THEN
        WRITE(1,12) ISIDE, ISEM, IATAF
      ELSE
        WRITE(1,13) ISIDE, ISEM, IATAF
      ENDIF
      RETURN
 11   FORMAT(´.PROC,XECUTE.´,/,
     &´APEGO,´,I1,´,´,I1,´,´,I1,´.´,/,
     &´EXIT,S.´,/,´REVERT.´)
 12   FORMAT(´.PROC,XECUTE.´,/,
     &´MIGO,´,I1,´,´,I1,´,´,I1,´.´,/,
     &´EXIT,S.´,/,´REVERT.´)
 13   FORMAT(´.PROC,XECUTE.´,/,
     &´LIGO,´,I1,´,´,I1,´,´,I1,´.´,/,
     &´EXIT,S.´,/,´REVERT.´)
      END
      SUBROUTINE DONE
CX  THIS SUBROUTINE BUILDS A FILE THAT WILL INFORM
CX   THE USER OF HIS FAILURE TO PROPERLY ACCESS THE
CX   THE FILES.
      WRITE(1,12)
      RETURN
 12   FORMAT('.PROC,XECUTE.',/,'CONNECT,OUTPUT.',/,
     &'COPY,KNUJ.',/,'RETURN,KNUJ.',/,
     &'RETURN,XECUTE,LGO.',/,'REVERT.',/,
     &'.DATA,KNUJ.',/,' YOU ARE OBVIOUSLY HAVING PROBLEMS.',/,
     &' PLEASE GET SOME HELP.',/,'.EOF')
      END
      SUBROUTINE INFORM
      COMMON/FILEDA/ICHOIS, ISEM, ISIDE, IATAF
      P IN  12
  12  FORMAT(´    PLEASE SIT BACK FOR A FEW MINUTES WHILE´,/,
     &´  YOUR FILES ARE ATTACHED.  THE PRINTOUT MAY BE´,/,
     &´  RATHER MESSY BUT IS, UNFORTUNATELY, NECESSARY.´,/,
     &´  DO NOT TOUCH ANY OF THE KEYS UNTIL THE PROGRAM´,/,
     &´  REQUESTS SOME INPUT.  THANK-YOU FOR YOUR COOPERATION.´)
      RETURN
      END
      SUBROUTINE INTER
      COMMON/FILEDA/ICHOIS, ISEM, ISIDE, IATAF
      COMMON/ITIMS/ITRY
CX  THE PURPOSE OF THIS SUBROUTINE IS TO INTERROGATE
CX     THE STUDENT TO DETERMINE HIS SEMINAR # (ISEM), SIDE
CX     (ISIDE, WHERE 1 => BLUE AND 2 => RED), AND ATAF
CX     IF APPLICABLE.
CX
      CHARACTER*4 INPT, IPASS(12), SIDE(2), IANS*1
      DATA SIDE/'BLUE', 'RED '/
      DATA IPASS/'PAS1','PAS2','PAS3','PAS4',
     &           'PAS5','PAS6','PAS7','PAS8',
     &           'PAS9','PAS0','PASA','PASB'/
CX  NOW INTERROGATE FOR SIDE, SEMINAR, AND ATAF
 10   PRINT *,' WHICH SIDE ARE YOU ON, RED OR BLUE ?  >'
      READ 23, INPT
      IF( INPT .NE. 'BLUE' .AND. INPT .NE. 'RED '
     &    .AND. INPT .NE. ' RED') THEN
        PRINT *,' ONLY ''BLUE'' OR ''RED '' ALLOWED '
        GO TO 10
      ENDIF
      ISIDE = 1
      IF( INPT .NE. 'BLUE' ) ISIDE = 2
CX  GET THE SEMINAR NUMBER
 20   PRINT *,' WHICH SEMINAR ARE YOU IN ( 1 OR 2 )?  >'
      READ *,ISEM
      IF( ISEM .NE. 1 .AND. ISEM .NE. 2 ) THEN
        PRINT *,' ONLY 1 OR 2 ALLOWED '
        GO TO 20
      ENDIF
      IF( ICHOIS .EQ. 2 ) THEN
CX  GET THE ATAF NUMBER
 30     PRINT *,' WHICH ATAF ( 2 OR 4 )?  >'
        READ *, IATAF
        IF( IATAF .NE. 2 .AND. IATAF .NE. 4 ) THEN
          PRINT *,' ONLY 2 OR 4 ALLOWED'
          GO TO 30
        ENDIF
      ENDIF
CX  CHECK IF INPUT DATA IS CORRECT SO FAR
      IF( ICHOIS .NE. 2 ) THEN
        IATAF = 0
        PRINT 12, SIDE(ISIDE), ISEM
      ELSE
        PRINT 22, SIDE(ISIDE), ISEM, IATAF
      ENDIF
      READ 13, IANS
      IF( IANS .NE. 'Y' ) GO TO 10
CX  CALCULATE THE PASSWORD INDEX FOR THIS COMBINATION
      INDEX = ISIDE * 6 + ISEM * 3 + IATAF/2 - 8
```

```
CX  ASK FOR THE PASSWORD
 40   PRINT *,´ INPUT THE PASSWORD PLEASE  >´
      READ 23, INPT
      IF( INPT .NE. IPASS(INDEX) ) THEN
        ITRY = ITRY + 1
        PRINT 32,INPT
        IF( ITRY .LT. 3 ) GO TO 40
      ELSE
        PRINT *,´ PASSWORD IS CORRECT.´
      ENDIF
      RETURN
 12   FORMAT(´ YOUR INPUTS ARE:´,/,´   SIDE       : ´,A4,/,
     &       ´    SEMINAR # : ´,I2,/,/,
     &´ IS THIS CORRECT ( Y OR N )  >´)
 22   FORMAT(´ YOUR INPUTS ARE:´,/,´   SIDE       : ´,A4,/,
     &       ´    SEMINAR # : ´,I2,/,´   ATAF #     : ´,I2,/,/,
     &´ IS THIS CORRECT ( Y OR N )  >´)
 32   FORMAT(´  SO SORRY, BUT ´,A4,´ IS NOT THE VALID PASSWORD.´)
 13   FORMAT(A1)
 23   FORMAT(A4)
      END
      SUBROUTINE STOPR
CX  THIS SUBROUTINE SAYS GOOD-BYE TO THE STUDENT.
CX
      WRITE(1,12)
      RETURN
 12   FORMAT(´.PROC,XECUTE.´,/,´CONNECT,OUTPUT.´,/,
     &´COPY,KNUJ.´,/,´RETURN,KNUJ.´,/,
     &´RETURN,XECUTE,LGO.´,/,´REVERT.´,/,
     &´.DATA,KNUJ.´,/,
     &´   YOU REALLY SHOULD NOT HAVE CHOSEN THIS OPTION.´,/,
     &´   IF THERE IS SOME QUESTION AS TO WHICH OPTION´,/,
     &´   YOU SHOULD HAVE CHOSEN, PLEASE CONTACT THE´,/,
     &´   GAME ADMINISTRATORS.  THANK-YOU.´,/,
     &´.EOF´)
      END
.EOF
```

Appendix B

TWX Library Routines Listings

```
      FUNCTION FLD( I, K, E )
CX*****************************************
CX*                                        *
CX*    THE PURPOSE OF THIS FUNCTION        *
CX* IS TO RETURN K CHARACTERS OF WORD E    *
CX* BEGINNING WITH POSITION (I+1).  THIS IS*
CX* THE FORTRAN 77 EQUIVALENT TO FUNCTION  *
CX* FLD AS DESCRIBED ON PAGE 6-7 OF THE    *
CX* HONEYWELL FORTRAN MANUAL.  HOWEVER,    *
CX* ONLY THE FIRST FORM OF FLD IS HANDLED  *
CX* BY THIS FUNCTION.  THE OTHER TYPE IS   *
CX* HANDLED BY FUNCTION FLD2.              *
CX*                                        *
CX********  C A U T I O N   ***************
CX*                                        *
CX*  E MUST BE OF CHARACTER*36 TYPE,       *
CX*  0 .LE. I .LE. 35                      *
CX*  1 .LE. K .LE. 36                      *
CX*                                        *
CX*****************************************
CX
      CHARACTER *(*) E
      J1 = I + 1
      J2 = J1 + K - 1
      FLD = FLOAT( ICH2N( E(J1:J2) ) )
      RETURN
      END
      FUNCTION ICH2N( ICH )
C
C****************************************C
C                                        C
CX   AUTHOR: ANTHONY WAISANEN            C
CX           CAPT        USAF            C
CX   DECEMBER 1981                       C
C                                        C
C      THE PURPOSE OF THIS FUNCTION      C
C   IS TO CONVERT A CHARACTER STRING TO  C
C   AN INTEGER.                          C
C                                        C
C  **********  C A U T I O N  ********** C
C                                        C
C    ICH HAD BETTER BE AN INTEGER        C
C                                        C
C****************************************C
C
      CHARACTER *(*) ICH
C
CX  FIND L (NUMBER OF CHARACTERS IN ICH)
C
      L = LEN( ICH )
C
CX  CONVERT ICH TO AN INTEGER
C
      ICH2N = 0
      DO 15 I = 1, L
C  MOVE ICH2N OVER 1 DECIMAL PLACE
      ICH2N = ICH2N * 10
      ICHADD = ICHAR( ICH(I:I) ) - 16
CX ADJUST IF OUTSIDE OF BOUNDS
      IF( ICHADD .GT. 9 ) ICHADD = 0
CX
      ICH2N = ICH2N + ICHADD
 15   CONTINUE
C
      RETURN
      END
      FUNCTION RAND( X )
CX******************************************
CX*                                        *
CX*   IN THE ORIGINAL PROGRAMS, "RAND" WAS *
CX* USED TO RETURN A UNIFORM RANDOM NUMBER *
CX* BETWEEN 0 AND 1.  THIS IS EQUIVALENT TO*
CX* THE CDC FUNCTION "RANF,"  THEREFORE,   *
CX* RANF WILL BE USED.                     *
CX*                                        *
CX******** C A U T I O N *******************
CX*                                        *
CX*   THIS FUNCTION MUST BE CHANGED IF A   *
CX* DIFFERENT MAIN FRAME IS TO BE USED.    *
CX*                                        *
CX******************************************
CX
      Y = .273 E 03
      Z = .1 E 01
      X = AMOD( Y*X, Z )
      RAND = X
      RETURN
      END
      INTEGER FUNCTION ISETSW( N )
CX***********************************
CX*                                 *
CX*   THE PURPOSE OF THIS FUNCTION  *
CX* IS TO SENSE THE VALUE OF THE NTH*
CX* BIT IN THE PROGRAM SWITCH WORD. *
CX* IN THE TWX PROGRAM, THE SWITCH  *
CX* WORD IS ONLY USED TO SWITCH FROM*
CX* THEATER LEVEL SIMULATION ( WHERE*
CX* BIT #35 = 0 ) TO CORPS LEVEL    *
CX* SIMULATION (BIT #35 = 1).  SINCE*
CX* ONLY THEATER LEVEL SIMULATIONS  *
CX* ARE DESIRED, THIS FUNCTION WILL *
CX* ALWAYS RETURN A 0.              *
CX*                                 *
CX***********************************
CX
      ISETSW = 0
      RETURN
      END
      INTEGER FUNCTION KOMPCH( IWORD1, N1, IWORD2, N2, M )
CX******************************************
CX*  THIS FUNCTION COMPARES M CHARACTERS OF IWORD1
CX* (BEGINNING WITH N1) WITH M CHARACTERS OF IWORD2
CX* (BEGINNING WITH N2).  KOMPCH = 0 IF THEY ARE
CX* EQUAL   1 IF THEY ARE NOT.
CX*
CX*********  C A U T I O N   ***************
CX*  IWORD1 AND IWORD2 MUST BE CHARACTER TYPE
CX*
CX*****************************************
      CHARACTER *(*) IWORD1, IWORD2
      LAST1 = N1 + M - 1
      LAST2 = N2 + M - 1
CX  INITIALIZE KOMPCH TO "FALSE"
      KOMPCH = 1
CX  COMPARE IWORD1 AND IWORD2
      IF( IWORD1(N1:LAST1) .EQ. IWORD2(N2:LAST2) ) KOMPCH = 0
      RETURN
      END
      LOGICAL FUNCTION CHKBIT(BIT,ARRAY,DIM)
C
C     CHECKS TO SEE IF BIT IS SET IN ARRAY - TRUE/FALSE
C     DEPENDING UPON BIT.
C     PIERCE......1980
CX    WAISANEN......1982
C
CX    INTEGER BIT,DIM,ARRAY,WORD,BIT1
CX    DIMENSION ARRAY(DIM)
      INTEGER BIT,DIM,WORD,BIT1
      CHARACTER*36 ARRAY(*)
C
      WORD=((BIT-1)/36)+1
      BIT1=MOD(BIT-1,36)
      CHKBIT=(FLD(BIT1,1,ARRAY(WORD)).EQ.1)
      RETURN
      END
      SUBROUTINE BCDASC( WORD1, WORD2, N )
CX*****************************************
CX*                                       *
CX*     THE PURPOSE OF THIS SUBROUTINE    *
CX* IS TO EQUATE WORD1 WITH WORD2.        *
CX* IT IS A NECESSARY SUBROUTINE WHEN     *
CX* USING A HONEYWELL COMPUTER SINCE NOT  *
CX* ALL VARIABLES ARE KEPT AS ASCII.  FOR *
CX* EXAMPLE, DATA STORED ON A DIRECT      *
CX* ACCESS FILE IS BCD WHEREAS DATA READ  *
CX* INTERACTIVELY IS ASCII.               *
CX*   SUCH IS NOT THE CASE WITH THE CDC   *
CX* SO, SINCE THE PROGRAMS USE THIS       *
CX* ROUTINE REGULARLY, THIS DUMMY ROUTINE *
CX* EXISTS RATHER THAN MAKING ALL CHANGES.*
CX*                                       *
CX********  C A U T I O N   ***************
CX*                                       *
CX*  WORD1 AND WORD2 MUST BE CHARACTER    *
CX* TYPE.  N IS A DUMMY ARGUMENT.  IN THE *
CX* ORIGINAL VERSION, IT DETERMINED THE   *
CX* NUMBER OF CHARACTERS TO BE CONVERTED. *
CX*                                       *
CX*****************************************
CX
      CHARACTER *(*) WORD1, WORD2
      WORD2 = WORD1
      RETURN
      END
      SUBROUTINE BLNKOU( BUFFR )
CX
CX******************************************
CX*                                        *
CX*     THE PURPOSE OF THIS SUBROUTINE     *
CX*  IS TO "BLANK-OUT" A CHARACTER STRING.*
CX*                                        *
CX********  C A U T I O N   ****************
CX*                                        *
CX*    BUFFR MUST BE CHARACTER TYPE.       *
CX*                                        *
CX******************************************
CX
      CHARACTER *(*) BUFFR
      WRITE(BUFFR,'(A1)') ' '
      RETURN
      END
      SUBROUTINE CALLSS( N )
CX*****************************************
CX
CX  THIS SUBROUTINE IS USED BY THE HONEYWELL
CX TO TRANSMIT JOB CONTROL LANGUAGE (JCL)
CX FROM AN EXECUTING PROGRAM TO THE SYSTEM.
CX THE FOLLOWING CODING IS JUST A DUMMY.
CX
CX*****************************************
CX
      CHARACTER *(*) N
      RETURN
      END
      SUBROUTINE CONCAT(STR1,N1,STR2,N2,N3)
C
C*****************************************************C
C                                                     C
CX   AUTHOR: ANTHONY WAISANEN                         C
CX          CAPT        USAF                          C
CX   DECEMBER 1981                                    C
C                                                     C
C      THE PURPOSE OF THIS SUBROUTINE IS CONCATINATE  C
C    A NUMBER OF CHARACTERS (N3) OF A CHARACTER       C
C    STRING (STR1) BEGINNING WITH THE N1 CHARACTER    C
C    WITH N3 CHARACTERS OF ANOTHER CHARACTER STRING   C
C    (STR2) BEGINNING WITH ITS N2 CHARACTER.          C
C                                                     C
C **********  C A U T I O N  **********               C
C                                                     C
C      N2 + N3 - 1  SHOULD NOT EXCEED THE NUMBER OF   C
C  CHARACTERS IS STR2 OR STRANGE THINGS WILL RESULT.  C
C                                                     C
C*****************************************************C
C
      CHARACTER * 1  STR1, STR2
      LAST1 = N1 + N3 - 1
      LAST2 = N2 + N3 - 1
C
      STR1(N1:LAST1) = STR2(N2:LAST2)
C
      RETURN
      END
      SUBROUTINE CREATE( LUD, ISIZE, N, ISTAT )
CX**************************************
CX*  THE PURPOSE OF THIS SUBROUTINE IS *
CX* TO CREATE A LOCAL DIRECT ACCESS    *
CX* FILE.  IT IS ESSENTIALLY THE EQUI- *
CX* VALENT OF THE HONEYWELL SUBROUTINE *
CX* OF THE SAME NAME.                  *
CX**************************************
CX
      OPEN( UNIT=LUD, RECL=ISIZE, IOSTAT=ISTAT, ERR=10,
     & ACCESS='DIRECT' )
      IF( ISTAT .NE. 0 ) THEN
 10     PRINT*,' TROUBLE IN CREATE WITH ISTAT =',ISTAT
      ENDIF
      RETURN
      END
      SUBROUTINE DATIM(DAYT,THYME)
C
C   **THIS ROUTINE RETURNS AN 8 CHARACTER DATE ('DAYT') FIELD AND A REAL
C    **VALUE OF TIME ('THYME') IN THE FORM OF 'MM/DD/YY' AND HH.XXXXX
C    **(THYME IS A REAL VARIABLE).
CX  **AUTHOR:  WAISANEN  2 DEC 81
CX  THIS SUBROUTINE IS COMPLETELY SYSTEM DEPENDENT
CX  BECAUSE OF THE WAY THE CURRENT DATE AND TIME
CX  ARE CALLED.  DEPENDING ON THE PARTICULAR SYSTEM
CX  BEING USED, THE CALLS TO "DATE" AND "TIME"
CX  SHOULD BE ALL THAT REQUIRE MODIFICATION.
C
CX  THIS SUBROUTINE HAS BEEN MODIFIED FOR THE
CX  CYBER/CDC MAINFRAME
      CHARACTER*10 DATE, CLOCK, DUM, DAYT*8
      REAL THYME
CX  FIRST, GET THE CURRENT DATE
      DUM = DATE(0)
      DAYT = DUM(2:9)
C  NOW FIND THE TIME
      DUM = CLOCK(0)
C  THE CHARACTER STRING 'DUM' IS OF THE FORM:
C      'HH.MM.SS.
C  AND MUST BE CONVERTED TO THE REAL VARIABLE 'THYME'
      THYME = FLOAT( ( ICHAR( DUM(2:2) ) - 16 ) * 10 +
     &               ( ICHAR( DUM(3:3) ) - 16 ) ) +
     &        FLOAT( ( ( ICHAR( DUM(5:5) ) - 16 ) * 10 +
     &                 ( ICHAR( DUM(6:6) ) - 16 ) ) * 60 +
     &                 ( ICHAR( DUM(8:8) ) - 16 ) * 10 +
     &                 ( ICHAR( DUM(9:9) ) - 16 ) ) / 3600.0
      RETURN
      END
      SUBROUTINE DETACH( IFILE, ISTAT )
CX*****************************************
CX*                                       *
CX*  THIS SUBROUTINE IS THE FORTRAN 77    *
CX* EQUIVALENT OF THE HONEYWELL ROUTINE   *
CX* OF THE SAME NAME.  UNLIKE THE HONEY-  *
CX* WELL ROUTINE, TAPE# IFILE IS NOT      *
CX* RETURNED BUT IS ONLY CLOSED.          *
CX*  IF ISTAT .NE. 0, THEN AN ERROR HAS   *
CX* OCCURED AND TAPE# IFILE MAY NOT BE    *
CX* PROPERLY CLOSED.                      *
CX*                                       *
CX********  C A U T I O N   ***************
CX*                                       *
CX*  THERE ARE NO CAUTIONS WITH THIS      *
CX* ROUTINE.                              *
CX*                                       *
CX*****************************************
      CLOSE( UNIT=IFILE, IOSTAT=ISTAT, ERR=10 )
CX ERR IS SPECIFIED TO FORCE THE PRINTING OF THE
CX  ERROR STATEMENT.
 10   IF( ISTAT .NE. 0 ) THEN
        PRINT 12, ISTAT, IFILE
      ENDIF
      RETURN
 12   FORMAT(' DETACH ERROR # ',I3,' WITH TAPE ',I2)
      END
      SUBROUTINE FLD2( I, K, E, N )
CX*****************************************
CX*                                       *
CX*     THE PURPOSE OF THIS SUBROUTINE     *
CX* IS TO SET K CHARACTERS OF WORD E TO # N*
CX* BEGINNING WITH POSITION (I+1).  THIS IS*
CX* THE FORTRAN 77 EQUIVALENT TO SECOND   *
CX* FORM OF THE FUNCTION FLD AS DESCRIBED *
CX* ON PAGE 6-7 OF THE HONEYWELL FORTRAN  *
CX* MANUAL.                               *
CX*                                       *
CX********  C A U T I O N   ***************
CX*                                       *
CX*  E MUST BE OF CHARACTER*36 TYPE,      *
CX*  0 .LE. I .LE. 35                     *
CX*  1 .LE. K .LE. 36                     *
CX*                                       *
CX*****************************************
CX
      CHARACTER *(*) E, NUM*1
      CALL N2CH( NUM, N )
      DO 15 J = 1, K
        IPOS = J + I
        E(IPOS:IPOS) = NUM
 15   CONTINUE
      RETURN
      END
```

```
CX*#RUNH *=;FKADY09/D070/FKADYLB/OBJECT/ILLEGAL(NOGO)
CX    SUBROUTINE ILLEGAL(IPOS)
      SUBROUTINE ILLEGA(IPOS)
C
C     WRITTEN BY CAPT. PIERCE......5 DEC 1979
CX    MODIFIED BY CAPT WAISANEN....MAR 1982
C
      CHARACTER*11 FMT
      DATA FMT/´(XXX,´´´  ´´)´/
C
      IF(.NOT.(IPOS.LE.77))GO TO 10
C       THEN
CX        ENCODE(FMT,100)IPOS
          WRITE(FMT,100)IPOS
          WRITE(6,FMT)
          GO TO 20
C       ELSE
 10       CONTINUE
          WRITE(6,200)IPOS
 20     CONTINUE
C     ENDIF
CX100  FORMAT(T2,I2)
 100  FORMAT(´(´,I2,´X,´´´  ´´)´)
 200  FORMAT(60X,´POSITION´,I3,´->´)
      RETURN
      END
```

```
CX  *#RUNH *=;FKADY01/SOURCE/TWXLIB/SLDSBUJ
CX  THE FOLLOWING SUBROUTINE HAS BEEN DETERMINED
CX  TO BE UNNECESSARY.  THEREFORE, IT HAS BEEN DELETED AND
CX  REFERENCES TO IT HAVE BEEN MODIFIED TO 'MSORTY'.
CX
CX    SUBROUTINE MBSORTY(BLF,ITAC,SRATE,SFACT,ACSORT)
C
C  BLALOCK:9 DEC 77
C  CHITWOOD:6 APR 81
C  CALCULATE THE NORMAL SORTIES
C
CX    ACSORT=AINT(BLF*FLOAT(ITAC)*SRATE*SFACT)
CX    RETURN
CX    END
C  *#RUNH *=;FKADY01/CSTR/TWXLIB/CLDSGUJ(NOGO)
      SUBROUTINE MSORTY(BLF,ITAC,SRATE,SFACT,ACSORT)
C
C  BLALOCK:9 DEC 77;CHITWOOD 6 APR 81
C  CALCULATE THE NORMAL SORTIES
C
               ACSORT=AINT(BLF*FLOAT(ITAC)*SRATE*SFACT)
      RETURN
      END
      SUBROUTINE N2CH( CHARA, I )
CX*****************************************
CX*                                       *
CX*     THE PURPOSE OF THIS SUBROUTINE    *
CX*  IS TO CONVERT AN INTEGER INTO AN     *
CX*  EQUIVALENT CHARACTER STRING.  IT IS  *
CX*  THE FORTRAN 77 EQUIVALENT OF AN      *
CX*  "ENCODE" STATEMENT.                  *
CX*                                       *
CX********  C A U T I O N   ***************
CX*                                       *
CX*  THE INTEGER ARGUMENT MUST BE INTEGER *
CX*   AND LESS THAN 10**15.               *
CX*                                       *
CX*****************************************
CX
      CHARACTER *(*) CHARA
      CHARA = ' '
      INTGR = I
      L = 1
      NP10 = 1
CX  IF I IS A ONE DIGIT NUMBER, SAVE CALCULATION
CX TIME BY SKIPPING.
      IF( INTGR .LT. 10 ) GO TO 10
CX  PERHAPS I IS A TWO DIGIT NUMBER?
      L = 2
      NP10 = 10
CX  SKIP IF THE NUMBER HAS ONLY TWO DIGITS.
      IF( INTGR .LT. 100 ) GO TO 10
CX  CALCULATE THE NUMBER OF DIGITS (L-1) ONLY
CX IF THE NUMBER IS GREATER THAN 99.
      L = IFIX( ALOG10( FLOAT( INTGR ) ) ) + 1
      NP10 = 10 ** (L-1)
CX
CX  NOW, PACK THE DIGITS INTO CHARA BEGINNING WITH
CX THE LEFT-MOST.
CX
 10   DO 15 IPOS = 1, L
      M =  INTGR / NP10
      CHARA(IPOS:IPOS) = CHAR( M + 16 )
      INTGR = INTGR - M * NP10
      NP10 = NP10 / 10
 15   CONTINUE
      RETURN
      END
      SUBROUTINE RANSIZ( IFILE, IRECSI, J )
CX*****************************************
CX*                                       *
CX*  THIS SUBROUTINE IS THE EQUIVALENT OF *
CX* A HONEYWELL FORTRAN ROUTINE OF THE    *
CX* SAME NAME.  ITS FUNCTION IS TO OPEN   *
CX* A DIRECT ACCESS FILE (TAPE# IFILE)    *
CX* WITH THE CORRECT NUMBER OF RECORDS    *
CX* PER RECORD (IRECSI).  J = 0 IMPLIES A *
CX* FORMATTED FORM OF FILE. J .NE. 0 IM-  *
CX* PLIES AN UNFORMATTED, DIRECT ACCESS   *
CX* FILE.                                 *
CX*                                       *
CX********  C A U T I O N   ***************
CX*   YOU HAD BETTER HAVE A LOCAL FILE BY *
CX* THE NAME OF "TAPE(IFILE)".            *
CX*                                       *
CX*****************************************
CX INITIALIZE ERROR INDICATOR ISTAT.
      ISTAT = 0
CX BEFORE OPENNING A FILE, IT MUST BE CLOSED
CX BUT ONLY CLOSE THE FILE IF RECL > 1
      IF( IRECSI .GT. 1 ) THEN
CX FILE HAS BEEN OPENNED BEFORE SO
      CLOSE( UNIT=IFILE, IOSTAT=ISTAT, ERR=10, STATUS='KEEP' )
      ENDIF
CX ERR IS SPECIFIED TO FORCE THE PRINTING OF THE ERROR STATEMENT.
 10   IF( ISTAT .EQ. 0 ) THEN
        IF( J .EQ. 0 ) THEN
          OPEN( UNIT=IFILE, IOSTAT=ISTAT, ERR=20, ACCESS='SEQUENTIAL',
     &          FORM='FORMATTED' )
        ELSE
          OPEN( UNIT=IFILE, IOSTAT=ISTAT, ERR=20, ACCESS='DIRECT',
     &          FORM='UNFORMATTED', RECL=IRECSI )
        ENDIF
 20     IF( ISTAT .NE. 0 ) THEN
          PRINT 22, ISTAT, IFILE
        ENDIF
      ELSE
        PRINT 12, ISTAT, IFILE
      ENDIF
      RETURN
 12   FORMAT(' CLOSE ERROR # ',I3,' WITH TAPE ',I2)
 22   FORMAT(' OPENNING ERROR # ',I3,' WITH TAPE ',I2)
      END
      SUBROUTINE READBU( N, CHAR )
CX
CX***********************************
CX   AUTHOR: ANTHONY WAISANEN           C
CX           CAPT        USAF           C
CX
CX  THE PURPOSE OF THIS SUBROUTINE
CX IS TO CHECK IF THE FIRST CHARACTER OF
CX CHAR IS A '0'.  IF IT IS,
CX N IS SET TO ZERO.  IF NOT, N IS
CX SET TO 3.
CX***********************************
CX
      CHARACTER *(*) CHAR
      N = 3
      IF( CHAR(1:1) .EQ. '0' ) N = 0
      RETURN
      END
      SUBROUTINE SETB(ISET,ISETD,MIN,IST,LST,ISTAT,INC,ON)
C     ROUTINE TO SET BITS IN ISET CORRESPONDING TO NUMBERIC INPUT.
C     YODIS.....13 JUL 78
C     PIERCE MODIFIED 5 DEC 79
CX    WAISANEN MODIFIED MAR L982
C
      LOGICAL ON
CX    DIMENSION ISET(ISETD)
      CHARACTER*36 ISET(*)
CX
C
      JNUM=0
      IF(ON)JNUM=1
      IF(.NOT.(IST.GT.LST))GO TO 90
C       THEN FIX
        M=IST
        IST=LST
        LST=M
   90 CONTINUE
C     ENDIF
      DO 110 NUM=IST,LST,INC
        IBIT=NUM-MIN
        JBIT=MOD(IBIT,36)
        JWRD=IBIT/36+1
        IF(JWRD.GT.ISETD)ISTAT=5
        IF(.NOT.(ISTAT.EQ.99))GO TO 100
C       THEN SET BITS.
CX        FLD(JBIT,1,ISET(JWRD))=JNUM
          CALL FLD2(JBIT,1,ISET(JWRD),JNUM)
  100   CONTINUE
C       ENDIF
  110 CONTINUE
C     ENDDO
      INC=1
      RETURN
      END
      SUBROUTINE SETBIT( IWORD, IBIT, ISW )
CX*****************************************
CX*                                       *
CX*     THE PURPOSE OF THIS SUBROUTINE    *
CX*  IS TO SET BIT # IBIT OF IWORD TO 1   *
CX*  IF ISW = 1 AND BIT # IBIT OF IWORD TO*
CX*  0 IF ISW = 0.                        *
CX*    1 .LE. IBIT .LE. 36.               *
CX*                                       *
CX********  C A U T I O N   ***************
CX*                                       *
CX*   IWORD MUST BE 36 CHARACTERS LONG.   *
CX*    ISW = 0 OR 1 AND 0 < IBIT < 37     *
CX*                                       *
CX*****************************************
CX
      CHARACTER*36 IWORD
CX  CHECK THE VALIDITY OF ISW AND IBIT
      IF(ISW .EQ. 0 .OR. ISW .EQ. 1 .AND.
     &   IBIT .GE. 1 .AND. IBIT .LE. 36) THEN
CX ISW AND IBIT IS VALID
CX SET CORRECT CHARACTER ('BIT') OF IWORD
          IF( ISW .EQ. 1 ) IWORD(IBIT:IBIT) = '1'
          IF( ISW .EQ. 0 ) IWORD(IBIT:IBIT) = '0'
      ELSE
CX  ERROR PROCESSING SECTION
          PRINT *,' ERROR IN SUBROUTINE SETBIT '
      ENDIF
      RETURN
      END
```

```
C *#RUNH *=;FKADY09/D070/FKADYLB/OBJECT/SEQIN(NOGO)
      SUBROUTINE SEQIN(BUF1,IP,ISET,IDIM,MAX,MIN,ISTAT)
C     ROUTINE TO DECODE A NUMERIC INPUT STRING OF THE FORM:
C       '1,2,4-6,10,...' AND SET CORRESPONDING BITS OF ISET.
C     YODIS.....14 JUL 78
C     MODIFIED 14 DEC 78      BY: LT HARBICK
C     BELLS AND WHISTLES ADDED 4 DEC 79 BY CAPT ROGER C PIERCE
CX    IMPROVED TO WORK ON A CDC MARCH L982 CAPT ANTHONY WAISANEN
C
      CHARACTER BUF1*80
C
      CHARACTER *15 CHARS
CX    DIMENSION ISET(IDIM)
      CHARACTER*36 ISET(*)
CX
      LOGICAL DASH,ON,COMMA
C
C
CX
      DATA CHARS/'0123456789-, AB'/
CX
      ISETD=IDIM
      DASH=.FALSE.
      NUM=0
      IBLANK=0
      IOFFSE=IP-1
      IP=IP-1
      ISTAT=99
      ON=.TRUE.
      INC=1
      COMMA=.TRUE.
C
C     DOUNTIL (IP.GE.80 .OR. ISTAT.NE.99)
  100 CONTINUE
      IP=IP+1
      ICHAR=16
      DO 110 ICH=1,15
        JCH=ICH
        IF(KOMPCH(BUF1,IP,CHARS,JCH,1).EQ.0) ICHAR = JCH
  110   CONTINUE
C     ENDDO
      IF(ICHAR.GE.1 .AND. ICHAR.LE.10)ICTYPE=1
      IF(ICHAR.GT.10)ICTYPE=ICHAR-9
C     CASE ENTRY
        GO TO (120,130,150,170,190,220,320)ICTYPE
C       CASE 1.  BUF1*IP IS A DIGIT.
  120 CONTINUE
      NUM=NUM*10+ICHAR-1
        COMMA=.FALSE.
        GO TO 330
```

```
C       CASE 2. BUF1*IP IS A DASH.
  130 CONTINUE
       IF(NUM.LT.MIN)ISTAT=4
       IF(NUM.GT.MAX)ISTAT=3
       IF(COMMA)ISTAT=6
       IF(KOMPCH(BUF1,IP+1,' ',1,1).EQ.0)ISTAT=6
       IF(IP.EQ.1)ISTAT=6
       IF(DASH)ISTAT=6
       IF(.NOT.(ISTAT.EQ.99))GO TO 140
C      THEN SET 1ST DO PARAM AND DASH FLAG.
        DASH=.TRUE.
        ISTNUM=NUM
        NUM=0
  140  CONTINUE
C     ENDIF
      COMMA=.FALSE.
      GO TO 330
C     CASE 3. BUF1*IP IS A COMMA.
  150 CONTINUE
      IF(NUM.LT.MIN)ISTAT=4
      IF(NUM.GT.MAX)ISTAT=3
      IF(.NOT.(NUM.EQ.0.AND..NOT.DASH))GO TO 155
C      THEN ZERO VALUE TERMINATES STRING
        ISTAT=0
        LENIN=IP-1
  155  CONTINUE
C     ENDIF
       IF(KOMPCH(BUF1,IP+1,' ',1,1).EQ.0)ISTAT=6
      IF(COMMA)ISTAT=6
      IF(IP.EQ.1)ISTAT=6
      IF(.NOT.(ISTAT.EQ.99))GO TO 160
C       THEN SET LAST DO PARAM AND SET BITS.
        LSTNUM=NUM
        NUM=0
        IF(.NOT.DASH)ISTNUM=LSTNUM
        DASH=.FALSE.
        CALL SETB(ISET,ISETD,MIN,ISTNUM,LSTNUM,ISTAT,INC,ON)
  160  CONTINUE
C     ENDIF
      COMMA=.TRUE.
      GO TO 330
C     CASE 4. BUF1*IP IS A SPACE.
  170 CONTINUE
      IBLANK=IBLANK+1
      IF(.NOT.(IBLANK.EQ.(IP-IOFFSE)))GO TO 175
C      THEN CHECK TO SEE IF LEADING BLANK LIMIT EXCEEDED
       IF(IBLANK.EQ.4)ISTAT=1
       GO TO 177
C      ELSE PROCESS AS END OF STRING
  175  CONTINUE
```

```
        IF(NUM.LT.MIN)ISTAT=4
        IF(NUM.GT.MAX)ISTAT=3
        IF(NUM.EQ.0.AND..NOT.DASH)ISTAT=0
        IF(.NOT.(ISTAT.EQ.99))GO TO 180
C         THEN SET LAST DO PARAM AND SET BITS.
          LSTNUM=NUM
          IF(.NOT.DASH)ISTNUM=LSTNUM
          DASH=.FALSE.
          CALL SETB(ISET,ISETD,MIN,ISTNUM,LSTNUM,ISTAT,INC,ON)
          ISTAT=0
 180    CONTINUE
C       ENDIF
 177  CONTINUE
C     ENDIF
      GO TO 330
C     CASE 5. BUF1*IP IS AN "A"
 190  CONTINUE
      IF(.NOT.(KOMPCH(BUF1,IP,'ALL',1,3).EQ.0))GO TO 200
C     THEN FILL ARRAY
          ISTNUM=MIN
          LSTNUM=MAX
          CALL SETB(ISET,ISETD,MIN,ISTNUM,LSTNUM,ISTAT,INC,ON)
          IP=IP+3
          IF(KOMPCH(BUF1,IP,',',1,1).NE.0) ISTAT=6
          IF(KOMPCH(BUF1,IP,' ',1,1).EQ.0) ISTAT=0
          GO TO 210
C       ELSE ILLEGAL CHARACTER
 200      CONTINUE
          ISTAT=2
 210    CONTINUE
C       ENDIF
      COMMA=.TRUE.
      GO TO 330
C     CASE 6. BUF1*IP IS A 'B'
 220  CONTINUE
      IF(.NOT.(KOMPCH(BUF1,IP,'BUT,',1,4).EQ.0.AND.COMMA))GO TO 230
C       THEN TURN OFF ON
          IF(.NOT.ON) ISTAT=8
          ON=.FALSE.
          IP=IP+3
          GO TO 310
C       ELSE CHECK FOR BY
 230      IF(.NOT.(KOMPCH(BUF1,IP,'BY',1,2).EQ.0.AND.COMMA))GO TO 290
C         THEN CHECK FOR INCREMENT
            INC=11
            DO 240 ICH=1,10
              JCH=ICH
              IF(KOMPCH(BUF1,IP+2,CHARS,JCH,1).EQ.0)INC=JCH-1
 240          CONTINUE
C           ENDDO
```

```
         IF(INC.EQ.0) INC=10
         IF(.NOT.(INC.EQ.11))GO TO 250
C        THEN ERROR
          ISTAT=7
          IP=IP+2
          GO TO 280
C        ELSE CHECK FOR COMMA
 250      CONTINUE
          IF(.NOT.(KOMPCH(BUF1,IP+3,´,´,1,1).EQ.0))GO TO 260
C          THEN OK
            IP=IP+3
            GO TO 270
C          ELSE DELIMETER ERROR
 260        CONTINUE
            ISTAT=6
 270       CONTINUE
C         ENDIF
280      CONTINUE
C        ENDIF
         GO TO 300
C        ELSE ILLEGAL CHARACTER
 290     CONTINUE
         ISTAT=2
         IF(.NOT.COMMA)ISTAT=6
 300     CONTINUE
C        ENDIF
 310    CONTINUE
C       ENDIF
        COMMA=.TRUE.
        GO TO 330
C       CASE 7. BUF1*IP IS AN ILLEGAL CHAR.
 320    CONTINUE
        ISTAT=2
 330  CONTINUE
C     ENDCASE.
      IF(.NOT.(IP.GE.80 .OR. ISTAT.NE.99))GO TO 100
C     ENDDO
      IF(ISTAT.EQ.4 .OR. ISTAT.EQ.3) IP=IP-1
      RETURN
      END
      SUBROUTINE SLITE( LITE )
CX*****************************************
CX*                                       *
CX*     THE PURPOSE OF THIS SUBROUTINE    *
CX* IS TO TURN ON A "SENSE LIGHT" WHICH IS*
CX* USED AS A FLAG IN THE MISSION INPUT   *
CX* PROGRAMS TO INDICATE THE PRESENCE OF  *
CX* ERRORS.                               *
CX*   IF THE VALUE OF LITE IS 0, THEN ALL *
CX* SENSE LIGHTS ARE TO BE TURNED OFF.    *
CX*   IF THE VALUE OF LITE IS NOT 0, THEN *
CX* SENSE LIGHT # LITE IS TURNED ON.      *
CX*   THIS SUBROUTINE IS INTENDED TO      *
CX* REPLACE SUBROUTINE SLITE AS DESCRIBED *
CX* ON PAGE 6-46 OF THE HONEYWELL FORTRAN *
CX* MANUAL.                               *
CX*                                       *
CX******** C A U T I O N  ****************
CX*                                       *
CX*  THE VALUE OF LITE MUST LIE BETWEEN   *
CX* 1 AND 36 (THE NUMBER OF CHARACTERS IN *
CX* THE SENSE LIGHT WORD, LIGHT).         *
CX*                                       *
CX*****************************************
CX
CX DEFINE THE SENSE LIGHT WORD, LIGHT
      COMMON /SENSE/LIGHT
      CHARACTER*36 LIGHT
CX CHECK THE VALIDITY OF LITE
      IF( LITE.GE.1 .AND. LITE.LE.35 ) THEN
CX LITE IS VALID SO OPERATE ACCORDINGLY
        IF( LITE .LE. 0 ) THEN
CX          TURN OFF ALL SENSE LIGHTS
          LIGHT = '0'
        ELSE
CX          TURN ON LIGHT # LITE
          LIGHT(LITE:LITE) = '1'
        ENDIF
      ELSE
CX  ERROR HAS OCCURRED
      PRINT 12,LITE
      ENDIF
      RETURN
 12   FORMAT(1X,' ERROR IN SUBROUTINE SLITE.  CALLED',
     &' WITH LITE = ',I3)
      END
      SUBROUTINE SLITET( LITE, K )
CX*****************************************
CX*     THE PURPOSE OF THIS SUBROUTINE    *
CX* IS TO SENSE IF A "SENSE LIGHT" IS ON. *
CX*  IF THE VALUE OF LITE IS 0, THEN AN   *
CX* ERROR CONDITION EXISTS.  K IS SET TO 0*
CX* AND AN ERROR MESSAGE IS PRINTED.      *
CX*   IF THE VALUE OF LITE IS NOT 0, THEN *
CX* SENSE LIGHT # LITE IS CHECKED.  IF IT *
CX* WAS ON (=1), K = 1.  IF IT WAS OFF    *
CX* (=0), K = 2.  THE LIGHT IS TURNED OFF *
CX* AND CONTROL RETURNS TO THE CALLING    *
CX* ROUTINE.                              *
CX*   THIS SUBROUTINE IS INTENDED TO      *
CX* REPLACE SUBROUTINE SLITET AS DESCRIBED*
CX* ON PAGE 6-46 OF THE HONEYWELL FORTRAN *
CX* MANUAL.                               *
CX********  C A U T I O N   ***************
CX*  THE VALUE OF LITE MUST LIE BETWEEN   *
CX* 1 AND 36 (THE NUMBER OF CHARACTERS IN *
CX* THE SENSE LIGHT WORD, LIGHT).         *
CX*****************************************
CX
CX DEFINE THE SENSE LIGHT WORD, LIGHT
      COMMON /SENSE/LIGHT
      CHARACTER*36 LIGHT
CX CHECK THE VALIDITY OF LITE
      IF( LITE.GE.1 .AND. LITE.LE.35 ) THEN
CX LITE IS VALID SO OPERATE ACCORDINGLY
CX  DEFINE A DEFAULT VALUE FOR K ("OFF" VALUE)
        K = 2
           IF( LIGHT(LITE:LITE) .EQ. '1' ) THEN
CX           LIGHT WAS ON
             K = 1
CX           NOW TURN IT OFF
             LIGHT(LITE:LITE) = '0'
CX         ELSE
CX           LIGHT WAS OFF SO USE DEFAULT VALUES.
           ENDIF
      ELSE
CX  ERROR HAS OCCURRED
        PRINT 12,LITE
        K = 0
      ENDIF
      RETURN
 12   FORMAT(1X,' ERROR IN SUBROUTINE SLITET.  CALLED',
     &' WITH LITE = ',I3)
      END
```

```
C  *#RUNH *=;FKADY01/CSTR/TWXLIB/CLDBLUJ(NOGO)
      SUBROUTINE TBLF(NAC,LOPT,BSTAT,TWXBLF)
C
C  BLALOCK: 17 APRIL 79
C  COMPUTE SORTIE SCALE FACTOR FROM BASE STAT AND LOAD FACTOR
C
            X=FLOAT(NAC)/(FLOAT(LOPT)+1.0E-20)
            X1=AMAX1(0.0,X-BSTAT)
            C1=0.5*X1
            C2=1.5*X1
            TWXBLF=(AMIN1(X,BSTAT)+C1*EXP(-C2))/AMAX1(1.0E-20,X)
      RETURN
CX    END
CX
CX  THE FOLLOWING SUBROUTINE HAS BEEN DETERMINED
CX  TO BE UNNECESSARY.  THEREFORE, IT HAS BEEN DELETED AND
CX  REFERENCES TO IT HAVE BEEN MODIFIED TO 'TBLF'.
C  *#RUNH *=;FKADY01/CSTR/TWXLIB/CLDBBUJ(NOGO,BCD)
CX    SUBROUTINE TBBLF(NAC,LOPT,BSTAT,TWXBLF)
C
C  BLALOCK:  17 APRIL 79
C  COMPUTE SORTIE SCALE FACTOR FROM BASE
C  STAT AND LOAD FACTOR
C
CX    X=FLOAT(NAC)/(FLOAT(LOPT)+1.0E-20)
CX    X1=AMAX1(0.0,X-BSTAT)
CX    C1=0.5*X1
CX    C2=1.5*X1
CX    TWXBLF=(AMIN1(X,BSTAT)+C1*EXP(-C2))/AMAX1(1.0-20,X)
CX    RETURN
      END
```

```
C  *#RUNH *=;FKADY01/CSTR/TWXLIB/CLDDBUJ(NOGO,BCD)
      SUBROUTINE TDATIM(DATETI)
C
C    **THIS ROUTINE RETURNS A 16 CHARACTER DATE TIME FIELD FOR
C    **PRINTOUT OF THE FORM 'YY MMM DD  HH:MM'
C    **AUTHOR:  ABBOTT     DATE: 12 DEC 77
CX  **MODIFIER:  WAISANEN  2 DEC 81
C
CX  THIS SUBROUTINE HAS BEEN MODIFIED FOR THE
CX  CYBER/CDC MAINFRAME
      CHARACTER DATETI*16,DATE*8,MON(12)*3,HOUR*2,MIN*2
      DATA MON/'JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP',
     &      'OCT','NOV','DEC'/
C
      CALL DATIM(DATE,TIME)
CX
CX  ORIGINAL CODING
CX    DECODE(DATE,500)MONTH
CX  EQUIVALENT FORTRAN 77 CODING
      READ(DATE,500)MONTH
CX
      DATETI='                '
      CALL CONCAT(DATETI,1,DATE,7,2)
      CALL CONCAT(DATETI,4,MON(MONTH),1,3)
      CALL CONCAT(DATETI,8,DATE,4,2)
CX
      IH=TIME
      IM=(TIME-FLOAT(IH))*60.0
CX
CX  ORIGINAL CODING
CX    ENCODE(HOUR,500)IH
CX    ENCODE(MIN,502)IM/10
CX    ENCODE(MIN,501)IM-IM/10*10
CX  EQUIVALENT FORTRAN 77 CODING
CX
      WRITE(HOUR,500)IH
      WRITE(MIN,502)IM/10
      WRITE(MIN,501)IM-IM/10*10
CX
      CALL CONCAT(DATETI,12,HOUR,1,2)
      CALL CONCAT(DATETI,14,':',1,1)
      CALL CONCAT(DATETI,15,MIN,1,2)
CX
      RETURN
  500 FORMAT(I2)
  501 FORMAT(T2,I1)
  502 FORMAT(T1,I1)
CX    END
CX
C  *#RUNH *=;FKADY01/CSTR/TWXLIB/CLDDTUJ(NOGO)
```

```
CX
CX  THE FOLLOWING SUBROUTINE HAS BEEN DETERMINED
CX  TO BE UNNECESSARY.  THEREFORE, IT HAS BEEN DELETED AND
CX  REFERENCES TO IT HAVE BEEN MODIFIED TO 'TDATIM'.
CX    SUBROUTINE TBDATIM(DATETIM)
C
C    **THIS ROUTINE RETURNS A 16 CHARACTER DATE TIME FIELD FOR
C    **PRINTOUT OF THE FORM 'YY MMM DD  HH:MM'
C    **AUTHOR:  ABBOTT     DATE: 12 DEC 77
C
CX    CHARACTER DATETIM*16,DATE*8,MON*3(12),HOUR*2,MIN*2
CX    DATA MON/'JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP',
CX   &      'OCT','NOV','DEC'/
C
CX    CALL DATIM(DATE,TIME)
CX    DECODE(DATE,500)MONTH
CX    DATETIM=' '
CX    CALL CONCAT(DATETIM,1,DATE,7,2)
CX    CALL CONCAT(DATETIM,4,MON(MONTH),1,3)
CX    CALL CONCAT(DATETIM,8,DATE,4,2)
CX    IH=TIME
CX    IM=(TIME-FLOAT(IH))*60.0
CX    ENCODE(HOUR,500)IH
CX    ENCODE(MIN,502)IM/10
CX    ENCODE(MIN,501)IM-IM/10*10
CX    CALL CONCAT(DATETIM,12,HOUR,1,2)
CX    CALL CONCAT(DATETIM,14,':',1,1)
CX    CALL CONCAT(DATETIM,15,MIN,1,2)
CX    RETURN
CX500  FORMAT(I2)
CX501  FORMAT(T2,I1)
CX502  FORMAT(T1,I1)
       END
```

```
      SUBROUTINE TERMNO( CHAR )
CX************************************
CX   AUTHOR: ANTHONY WAISANEN            C
CX           CAPT        USAF            C
CX
CX  THE PURPOSE OF THIS SUBROUTINE
CX ORIGINALLY WAS TO RETURN THE 2 CHARACTER
CX "NAME" OF THE TERMINAL BEING USED.
CX  THE ORIGINAL SUBROUTINE WAS A
CX HONEYWELL SYSTEM ROUTINE.  THERE IS
CX NO ANSII STANDARD ROUTINE.
CX THIS, THEN, IS A DUMMY.
CX************************************
CX
      CHARACTER*2 CHAR
      CHAR = '77'
      RETURN
      END
```

```
CX  *#RUNH *=;FKADY01/CSTR/TWXLIB/CLDLTUJ(NOGO,BCD)
CX     SUBROUTINE TLETTER(LINES,LETTERS,STRING,IFC)
       SUBROUTINE TLETTE(LINES,LETTER,STRING,IFC)
C
C     BLALOCK: MODIFIED 10 DEC 77
C      NUMERIC - CHARACTER CONVERSION & PRINT
C
C        /
       IMPLICIT INTEGER(A-Z)
       DIMENSION MAP(100),STRING(18,LINES)
       CHARACTER*1 X,INDX(32)
       CHARACTER*7 SYMB(100)
       CHARACTER*5 BINY(32),PRNT(7,18)
       DATA INDX/'0','1','2','3','4','5','6','7','8','9','A','B','C',
     &        'D','E','F','G','H','I','J','K','L','M','N','O','P',
     &        'Q','R','S','T','U','V'/
       DATA BINY/'     ','*    ',' *   ','**   ','  *  ','* *  ',' **  ',
     &        '***  ','   * ','*  * ',' * * ','** * ','  ** ','* ** ',
     &        ' *** ','**** ','    *','*   *',' *  *','**  *','  * *',
     &        '* * *',' ** *','*** *','   **','*  **',' * **','** **',
     &        '  ***','* ***',' ****','*****'/
       DATA SYMB/'EHPLJHE','464444E','EHGE11V','EHGCGHE','8CA9V88',
     &'V1FGGHE','C21FHHE','VG84222','EHHEHHE','EHHUG86','EHHVHHH',
     &'FIIEIIF','EH111HE','FIIIIIF','V11F11V','V11F111','U11THHE',
     &'HHHVHHH','E44444E','GGGGGHE','H95359H','111111V','HRLLHHH',
     &'HJLPHHH','VHHHHHV','FHHF111','EHHHL9M','FHHF59H','EH248HE',
     &'V444444','HHHHHHE','HHHAA44','HHHLLLA','HHA4AHH','HHA4444',
     &'VG8421V','0AR0RA0','EHGMLD3','2552L9M','00V0V00','04EVE40',
     &'3J842P0','4AH0000','EH84404','044V440','01248G0','0G84210',
     &'000V000','0000066','0006642','6606642','0660660','7444447',
     &'S44444S','G84248G','1248421','1244421','G84448G','8420000',
     &'6666066','4444444','VVVVVVV','0000000','AA00000','EL5EKLE',
     &'SOM9996','EHHE4E4','8088896','1195359','4444444','00ALLLL',
     &'00DJHHH','00EHHHE','00FHF11','00E9E80','00DJ111','00E1687',
     &'22F22IC','009999M','00HHLLA','00HA4AH','00HHUGE','00HHHA4',
     &'00V842V','006999M','11DJHJD','00EH1HE','GGMPHPM','00EHV1E',
     &'04U4444','00E9E86','11DJHHH','4044444','00000V','002V200',
     &'8442448','2448442','00QD000',2*'       '/
       DATA MAP/ 60, 61, 62, 63, 64, 65, 66, 67, 70, 71,101,102,103,104,
     &105, 106,107,110,111,112,113,114,115,116,117,120,121,122,123,124,
     &     125,126,127,130,131,132, 43,100, 46, 75, 52, 45,136, 77, 53,
     &     134, 57, 55, 56, 54, 73, 72,135,133, 74, 76, 51, 50, 47, 41,
     &     174,200, 40, 42, 44,201,202,152,153,154,155,156,157,160,161,
     &     162,163,164,165,167,170,171,166,172,141,142,143,144,145,146,
     &     147,150,151,137,140,173,175,176, 2*0/
       DATA X/' '/,D1FAUL/63/,D2FAUL/1/
C
       DO 400 L=1,LINES
         DO 300 N=1,LETTER
           SEED=D1FAUL
         DO 100 K=1,100
           IF(STRING(N,L).EQ.MAP(K)) SEED=K
100     CONTINUE
         DO 200 H=1,7
           HX=H
           CALL CONCAT(X,1,SYMB(SEED),HX,1)
           MX=D2FAUL
           DO 150 M=1,32
             IF(X.EQ.INDX(M)) MX=M
150        CONTINUE
           PRNT(H,N)=BINY(MX)
200      CONTINUE
300    CONTINUE
        DO 350 H=1,7
          WRITE(IFC,600) (PRNT(H,N),N=1,LETTER)
350    CONTINUE
        WRITE(IFC,601)
400   CONTINUE
      RETURN
600   FORMAT(1X,18(A5,2X))
601   FORMAT(//)
      END
      SUBROUTINE UPRCAS( N, M )
CX*****************************************
CX*                                       *
CX*  THIS IS A DUMMY SUBROUTINE.  IN THE  *
CX* HONEYWELL SYSTEM, IT CONVERTS A LOWER *
CX* CASE STRING OF M CHARACTERS, N, INTO  *
CX* AN EQUIVALENT UPPERCASE STRING, ALSO  *
CX* N.  SINCE ASCII ONLY USES UPPERCASE,  *
CX* THIS SUBROUTINE HAS NO FUNCTION OTHER *
CX* THAN TO INDICATE WHERE THE ORIGINAL   *
CX* TWX DATA HAD TO BE CONVERTED.         *
CX*                                       *
CX********  C A U T I O N   ***************
CX*                                       *
CX*  THERE ARE NO CAUTIONS ASSOCIATED WITH*
CX* THIS SUBROUTINE.                      *
CX*                                       *
CX*****************************************
      CHARACTER *(*) N
      RETURN
      END
```

Appendix C

Segmentation Procedure File Listings

```
.PROC,APESEG,FYLE=APE.
.*  THIS PROCEDURE FILE CREATES A SEGMENTED OBJECT
.* DECK AND CATALOGS IT AS "APELGO."
.* THE SOURCE DECK "FYLE" MUST BE THE SOURCE DECK
.* FOR THE AAFCE PLANNING PROGRAM.
MAP,OFF.
RETURN,KNUJ.
ATTACH,KNUJ,APELGO,ID=T800855,PW=FOX.
.*  SKIP IF NOT CATALOGED
SKIP,NONE.
EXIT,S.
ENDIF,NONE.
.*  CONTINUE FROM HERE
REWIND,FYLE.
FTN5,I=FYLE,LO=0,B=APEB.
.* CHECK FOR TWXLIB PRESENCE
IFE,FILE(ZZZZZLW,LO.OR.PF).NE.1,GET1.
.* NOT HERE, SO MUST GET IT
ATTACH,ZZZZZLW,TWXLIB,ID=T800855,CY=1.
ENDIF,GET1.
.* DECLARE IT AND THE UEDIT LIBRARY
LIBRARY(ZZZZZLW,ZZZZZLA)
RETURN,SEGLGO.
REQUEST,SEGLGO,*PF.
SEGLOAD(I=DIRS,B=SEGLGO)
LOAD(APEB)
NOGO.
.* RECALL THE LIBRARIES
LIBRARY(ZZZZZLA,ZZZZZLB)
CATALOG,SEGLGO,APELGO,ID=T800855,XR=FOX,PW=FOX.
PURGE,KNUJ.
RENAME,SEGLGO,CY=1.
RETURN,KNUJ,SEGLGO.
RETURN,DIRS,APEB.
.* NO MATTER WHAT
EXIT,S.
LIBRARY(ZZZZZLA,ZZZZZLB)
RETURN,DIRS,APEB.
REVERT.
```

```
.DATA,DIRS.
*
* MAJOR TREE APE
      TREE       APE-(IPT,OPTT,RAPUP)
      INCLUDE    UTLLIN,BLKDAT
*
* SUBTREE IPT (FILE INITIALIZATION)
IPT   TREE     PI-(PIRANS,PIDS-PIDAYS,PIAC,PIAB,PIMU,PILREC)
PI   INCLUDE    UTBLF
*
* SUBTREE OPTT ( OPTION HANDLING ROUTINES )
OPTT  TREE       OPTN-(DPT,IPRC1,IPAG1,IPRS1,IPIV1,LAT)
OPTN  INCLUDE    UTIMEO,UTLLIN,UTMLEF
*
*    SUB-SUBTREE DPT ( DISPLAY ROUTINES )
DPT   TREE       DP-(DP12,DPAG,DPTS-DPTSOT,DP13,DPDR)
DP   INCLUDE    DPACIN,DPSETB,DPABNO
*
*     SUBTREE OF DPT
DP12  TREE       DPBD-(DP2,DPBDOT)
DP2   TREE       DPACIN-DPACNA
DPACIN INCLUDE    DPSETB
*
*    ANOTHER SUBTREE OF DPT
DP13  TREE       DPBL-(DP3,DPBLOT)
DP3   TREE       DPLGIN-DPLGNA
DPLGIN   INCLUDE    DPSETB
*
*    SUB-SUBTREE IPRC1
IPRC1 TREE       IPRC-(IPRCED-IPRCHC,IPRCUP)
IPRC INCLUDE    IPCHEC,UTNUM,UTIMEO,UTLLIN,UTMLEF
*
*    SUB-SUBTREE IPAG1
IPAG1 TREE       IPAG-(IPAGED,IPAGUP)
IPAG INCLUDE    UTBLF,UTNUM,UTIMEO,UTLLIN,UTMLEF,IPCHEC
*
*    SUB-SUBTREE IPRS1
IPRS1 TREE       IPRS-(IPRSED,IPRSUP)
IPRS INCLUDE    IPCHEC,UTNUM,UTIMEO,UTLLIN,UTMLEF
*
*    SUB-SUBTREE IPIV1
IPIV1 TREE       IPIV-(IPIVED,IPIVCR,IPIVUP)
IPIV INCLUDE    IPCHEC,UTNUM,UTIMEO,UTLLIN,UTMLEF
*
*    SUB-SUBTREE LAT
LAT   TREE       LA-(LAATAF,LA12,LA22)
LA    INCLUDE    LAINMS,LAWXSC,LAOTCA
*
*    SUBTREE OF LAT
LA12  TREE       LAIN-(LANXQM,LA1)
```

```
*
*    SUBTREE OF LA12
LA1   TREE       LAINED-(LAINAC,LAINSR)
*
*    SUBTREE OF LAT
LA22  TREE       LAOT-(LAOTGR,LA2,LAOTPR)
*
*    SUBTREE OF LA22
LA2   TREE       LAOTAB-LAOTNO
LAOTAB   INCLUDE    LAOTSE
*
* COMMON BLOCK DECLARATION
          COMMON     ABDATA,ABLF,ACDATA,CTRL,IO,IOCH,
,LAINB,LAINBC,LAIOBU,LAOTG,LAOTMU,LGDATA,LSTGD,MSNS,
,NAMES,POLSPR
*
*  APE DECLARED TO BE ENTRY POINT
      END        APE
```

```
.PROC,LISEG,FYLE=LI.
RETURN,KNUJ.
ATTACH,KNUJ,LILGO,ID=T800855,PW=FOX.
SKIP,WHERE.
EXIT,S.
ENDIF,WHERE.
REWIND,FYLE.
FTN5,I=FYLE,B=LIB,LO=0,DB=0.
.* CHECK FOR TWXLIB PRESENCE
IFE,FILE(ZZZZZLW,LO.OR.PF).NE.1,GET1.
.* NOT HERE, SO MUST GET IT
ATTACH,ZZZZZLW,TWXLIB,ID=T800855,CY=1.
ENDIF,GET1.
.* DECLARE IT AND THE UEDIT LIBRARY
LIBRARY(ZZZZZLW,ZZZZZLA)
RETURN,SEGLGO.
REQUEST,SEGLGO,*PF.
SEGLOAD(I=DIRS,B=SEGLGO)
LOAD(LIB)
NOGO.
CATALOG,SEGLGO,LILGO,ID=T800855,PW=FOX,XR=FOX.
PURGE,KNUJ.
SKIP,NO.
EXIT,S.
ENDIF,NO.
RENAME,SEGLGO,CY=1.
RETURN,DIRS,XECUTE,KNUJ,SEGLGO.
.* RECALL THE LIBRARIES
LIBRARY(ZZZZZLA,ZZZZZLB)
.* NO MATTER WHAT
EXIT,S.
RETURN,DIRS,XECUTE.
LIBRARY(ZZZZZLA,ZZZZZLB)
REVERT.
```

```
.DATA,DIRS.
          TREE      LI-(LINKA,OPTREE,WURAP)
*
 LI       INCLUDE   BLKDAT
*
 LINKA    TREE      PI-(PIAC,PIATCH,PILI,
,PILO-(PILOPS-PILOCO),
,PILOSE,PILOSI,PINIT,PIARST)
*
 PINIT    INCLUDE   DATIM,FLD2,FP,RANSIZ,UTINAC,UTINCH,
,UTINDA,UTINOR,UTINTG,UTINUT
*
 OPTREE   TREE      OPTN-(LINKB,LINKC,LINKD,LINKE)
*
 OPTN     INCLUDE   UTHELP
*
 LINKB    TREE      DISPLA-(DISPDA,DISPPR,DISPUN-(DISPWX-DISPSV),
,DISPMI,DISPAC,
,DISPTG-(DISPTU,DISPTC,DISPTQ),
,DISPAM,DISPAD,DISPRE)
*
 DISPLA   INCLUDE   UTHELP
*
 DISPAC   INCLUDE   CONCAT,DATIM,DISPWA,UTACVD,UTTOKE
*
 DISPAD   INCLUDE   DATIM,DISPWA
*
 DISPAM   INCLUDE   DATIM,DISPWA,UTHELP
*
 DISPDA   INCLUDE   DATIM,DISPWL,UTHELP
*
 DISPMI   INCLUDE   DATIM,DISPWL,UTHELP
*
 DISPRE   INCLUDE   CONCAT,DATIM,DISPWA,DISPWL,FLD,
,FLD2,ILLEGA,SEQIN,UTHASH,UTHELP
*
 DISPTG   INCLUDE   CHKBIT,CONCAT,DATIM,DISPWA
*
 DISPTC   INCLUDE   CONCAT,ILLEGA,SEQIN
*
 DISPTQ   INCLUDE   CONCAT,ILLEGA,UTCOIN,UTTOKE
*
 DISPTU   INCLUDE   CONCAT,ILLEGA,SEQIN
*
 DISPWX   INCLUDE   DISPWL
*
 LINKC    TREE      MODIFY-(MODDEL-MODDCH,CHGTREE)
*
 MODIFY   INCLUDE   ILLEGA
 MODDEL   INCLUDE   FLD2
```

```
*
*
 CHGTREE TREE      MODCHG-(MODWT1,MODWT2,MODWT3,
,MODACF,MODCOR,MODCYC,MODDAY,MODPRI,MODQNT,
,MODTGT,MODTUT,MODFUT,MODURQ,MODLLQ)
*
 MODACF  INCLUDE   CONCAT,UTACVD,UTTOKE
*
 MODCOR  INCLUDE   CONCAT,UTCOIN,UTTOKE
*
 MODCYC  INCLUDE   UTDAVD
*
 MODDAY  INCLUDE   CONCAT,UTCOIN,UTDAVD,UTTOKE
*
 MODPRI  INCLUDE   CONCAT,UTCOIN,UTTOKE
*
 MODQNT  INCLUDE   UTQUAN
*
 MODTGT  INCLUDE   CONCAT,FLD,UTCOIN,UTTOKE
*
 MODTUT  INCLUDE   CONCAT,UTCOIN,UTERR,UTTOKE
*
 MODFUT  INCLUDE   CONCAT,UTCOIN,UTDAVD,UTERR,UTTGVD,UTTOKE
*
 MODURQ  INCLUDE   CONCAT,UTCOIN,UTTOKE
*
 MODLLQ  INCLUDE   CONCAT,UTCOIN,UTTOKE
*
 LINKD   TREE      LANDIN-(LAATK,LADEF,LALAY,LAMOV,LAREI,LAWIT)
*
 LAATK   INCLUDE   CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
,UTINCH,UTTGVD,UTTOKE
*
 LADEF   INCLUDE   CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
,UTINCH,UTTGVD,UTTOKE
*
 LALAY   INCLUDE   CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
,UTINCH,UTQUAN,UTTGVD,UTTOKE
*
 LAMOV   INCLUDE   CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
,UTINCH,UTTGVD,UTTOKE
*
 LAREI   INCLUDE   CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
,UTINCH,UTQUAN,UTTGVD,UTTOKE
*
```

```
 LAWIT   INCLUDE   CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
,UTINCH,UTQUAN,UTTGVD,UTTOKE
*
 LINKE   TREE      AIRIN-(AIRPRI,AIRRES,AIRREC,AIRBAI,AIRCAS,AICYVD)
 AIRIN   INCLUDE   UTHELP
*
 AIRPRI  INCLUDE   CHKBIT,CONCAT,ILLEGA,SEQIN,UTHELP
*
 AIRRES  INCLUDE   AICOVD,AIPIVD,CONCAT,UTABRT,UTACVD,UTCOIN,
,UTHASH,UTINCH,UTQUAN,UTTOKE
*
 AIRREC  INCLUDE   AIQDVD,AIPIVD,AITGVD,CONCAT,FLD,UTABRT,
,UTACVD,UTCOIN,UTHASH,UTINCH,UTTOKE
*
 AIRBAI  INCLUDE   AIPIVD,AITGVD,CONCAT,FLD,UTABRT,
,UTACVD,UTCOIN,UTHASH,UTINCH,UTQUAN,UTTOKE
*
 AIRCAS  INCLUDE   AICOVD,AIPIVD,CONCAT,UTABRT,UTACVD,UTCOIN,
,UTHASH,UTINCH,UTQUAN,UTTOKE
*
 AICYVD  INCLUDE   CONCAT
*
         COMMON
         END       LI
```

```
.PROC,MISEG,FYLE=MI,MAP=OFF.
RETURN,KNUJ,LGO.
ATTACH,KNUJ,MILGO,ID=T800855,PW=FOX.
.* SKIP IF NOT CATALOGED
SKIP,NONE.
EXIT,S.
ENDIF,NONE.
.* CONTINUE FROM HERE
REWIND,FYLE.
#MAP=MAP.
FTN5,I=FYLE,B=MIB,LO=0,DB=0.
.* CHECK FOR TWXLIB PRESENCE
IFE,FILE(ZZZZZLW,LO.OR.PF).NE.1,GET1.
.* NOT HERE, SO MUST GET IT
ATTACH,ZZZZZLW,TWXLIB,ID=T800855,CY=1.
ENDIF,GET1.
.* DECLARE IT AND THE UEDIT LIBRARY
LIBRARY(ZZZZZLW,ZZZZZLA)
RETURN,SEGLGO.
REQUEST,SEGLGO,*PF.
SEGLOAD(I=DIRS,B=SEGLGO)
LOAD(MIB)
NOGO.
CATALOG,SEGLGO,MILGO,ID=T800855,PW=FOX,XR=FOX.
PURGE,KNUJ.
.* SKIP TO "EXIT,S." STATEMENT IF FILE WAS NOT ATTACHED.
RENAME,SEGLGO,CY=1.
.* ALWAYS GO THROUGH THESE STATEMENTS
SKIP,NONE2.
EXIT,S.
ENDIF,NONE2.
RETURN,MIB,DIRS,XECUTE,KNUJ,SEGLGO.
.* RECALL THE LIBRARIES
LIBRARY(ZZZZZLA,ZZZZZLB)
.* NO MATTER WHAT
EXIT,S.
RETURN,MIB,DIRS,XECUTE.
LIBRARY(ZZZZZLA,ZZZZZLB)
REVERT.
```

```
.DATA,DIRS.
*
* MAIN TREE
          TREE      MI-(ASKIT,CHANGE,CYCINPT,CYFMT1T,
,CYFMT2T,CYFMT3T,CYFMT4T,CYPRP1,DAYINP,
,DAYRAP,DELETE,ENTERT,INIT,INTERMT,
,MODIFYT,PRINTI,RESEQ,RESTAR)
*
MI     INCLUDE    CYOUT,CYPRP3,DAYOUT,ENCOD1,ENCOD2,ENCOD3,
,ENCOD4,PRNT1,SLITE,SLITET
*
* TREE FOR LINK A ( AKA "PI" )
ENTERT    TREE       ENTER-(INTWX,SETREC)
*
ENTER     INCLUDE    DETACH,PRNT1
INTWX     INCLUDE    CHKSYS,RANSIZ
SETREC    INCLUDE    RANSIZ
*
* TREE FOR LINK B ( AKA "RT" )
INTERMT   TREE       INTERM-YNC
*
INTERM    INCLUDE    FLD2,PRNT1
RESTAR    INCLUDE    ARRAYS,BCDASC,CALCSO,CYPRP3,SLITE
*
* TREE FOR LINK C ( AKA "DI" )
CYCINPT   TREE       CYCINP-(CYCPRE-(INACMA,INTGTT))
*
CYCINP    INCLUDE    CONCAT,ENCOD1,ENCOD2,ENCOD3,ENCOD4,
,PRNT1,RPRNT
INACMA    INCLUDE    FLD2,SETBIT
INTGTT    INCLUDE    FLD2
CYCPRE    INCLUDE    FLD,SLITE,SLITET
CYPRP1    INCLUDE    SLITE,SLITET
PRINTI    INCLUDE    PRNT1
*
* TREE FOR LINK D ( AKA "F1" )
CYFMT1T   TREE       CYFMT1-(FMT1A-(CKNUM1,EDIT1T))
*
CYFMT1    INCLUDE    FLD,PRNT1,RPRNT
FMT1A    INCLUDE    BLNKOU,CKCMA1,CONCAT,DETACH,
,ERR1,GREASE,SLITE,SLITET,UPDATE,UPRCAS
*
EDIT1T    TREE       EDIT1-(EDITA1,DECOD1,EDITB1,
,EDITC1,EDITD1)
EDIT1     INCLUDE    CONCAT,ERR1,PRNT1,SLITE,
,SLITET,UPRCAS
DECOD1    INCLUDE    SLITE
EDITA1    INCLUDE    ICHKAC,SLITE,SLITET
EDITB1    INCLUDE    CHKACM,FLD2,SLITE,SLITET,SETBIT
EDITC1    INCLUDE    SLITE,UPRCAS
```

```
EDITD1     INCLUDE     CHKTGH
*
* TREE FOR LINK E ( AKA "F2" )
CYFMT2T    TREE        CYFMT2-(FMT2A-(CKNUM2,EDIT4T))
*
CYFMT2     INCLUDE     FLD,PRNT1,RPRNT
FMT2A     INCLUDE     BLNKOU,CKCMA1,CONCAT,DETACH,ERR1,GREASE,
,SLITE,SLITET,UPDATE,UPRCAS
*
EDIT4T     TREE        EDIT4-(DECOD2,EDITC2)
EDIT4      INCLUDE     EDITA2,EDITB2,FLD2,ICHKAC,SLITE,
,SLITET,CHKACM,SETBIT
EDITC2     INCLUDE     CHKTGH,UPRCAS
*
* TREE FOR LINK F ( AKA "F3" )
CYFMT3T    TREE        CYFMT3-(FMT3A-(CKNUM3,DECOD3,EDITC3))
*
CYFMT3   INCLUDE    FLD,PRNT1,RPRNT
FMT3A  INCLUDE     BLNKOU,CHKACM,CKCMA1,CONCAT,EDITA2,EDITB2,
,ERR1,FLD2,GREASE,ICHKAC,RPRNT,PRNT1,SETBIT,SLITE,
,SLITET,UPDATE,UPRCAS
*
* TREE FOR LINK G ( AKA "F4" )
CYFMT4T    TREE        CYFMT4-(FMT4A-(CKNUM4,DECOD4,EDITC4T))
*
CYFMT4     INCLUDE     FLD,PRNT1,RPRNT
FMT4A  INCLUDE     BLNKOU,CHKACM,CKCMA1,CONCAT,EDITA2,EDITB2,
,ERR1,FLD2,ICHKAC,GREASE,PRNT1,RPRNT,SETBIT,SLITE,SLITET,
,UPDATE,UPRCAS
EDITC4T    TREE        EDITC4-(EDITD4,EDITF4)
EDITC4     INCLUDE     CHKTGH,ERR1,FLD,PRNT1,SLITE,SLITET
*
* TREE FOR LINK H ( AKA "CH" )
CHANGE     INCLUDE     FILESC,PRNT1,SLITE,UPRCAS
RESEQ      INCLUDE     SHUFFL
DELETE     INCLUDE     ARRAYS,CONCAT,EDIT1C,FILESC,PRNT1,SLITE
*
* TREE FOR LINK I ( AKA "PR" )
ASKIT      INCLUDE     PRNT1
*
* TREE FOR LINK K ( AKA "MO" )
MODIFYT    TREE    MODIFY-(FMAT1R,FMAT2R,FMAT3R,FMAT4R,FMAT1W,
,FMAT2W,FMAT3W,FMAT4W)
*
MODIFY     INCLUDE     ARRAYS,CONCAT,EDIT1C,PRNT1,SLITE,UPRCAS
*
* TREE FOR LINK L ( AKA "WU" )
INIT       INCLUDE     BCDASC,CALCSO,CALLSS,TERMNO
DAYRAP     INCLUDE     DETACH,GREASE
*
```

```
* FINISH OF SEGMENTATION
        COMMON      ACDATA,CONTRC,CONTRO,CYCLC,CYCLE,CYCLEC,
,FILEDA,FLAGS,READIN,READIC,SENSE,SORDAT
          END         MI
.EOF
```

Appendix D

SETUP Listing

```
.PROC,SETUP.
.*  THE PURPOSE OF THIS PROCEDURE FILE IS TO
.* BUILD ALL LIBRARIES AND OBJECT DECKS NECESSARY
.* FOR THE TWX.  THE LIBRARIES WILL BE CREATED
.* INTERACTIVELY WHILE THE OBJECT DECKS WILL
.* BATCHED.
.*  FIRST, THE JCL FOR THE BATCH JOB
.* IS SENT TO THE INPUT QUEUE.
RETURN,KNUJ.
REQUEST,KNUJ,*Q.
REWIND,JOB.
COPY,JOB,KNUJ.
BATCH,KNUJ,INPUT,HERE.
RETURN,JOB.
.*  NOW, THE LIBRARIES ARE BUILT AND
.* FILLED WITH THE NECESSARY DATA FILES.
BUILD,ID=BACKUP.
BUILD,ID=BATCHIN.
BUILD,ID=MASTER.
BUILD,ID=PRINT.
GO,1.
GO,2.
GO,3.
GET,CAKM,ID=TWXDATA.
SAVE,CAKM=CAKW,ID=MASTER.
RETURN,CAKM.
GET,CCAM,ID=TWXDATA.
SAVE,CCAM=CCAW1,ID=BATCHIN.
RETURN,CCAM.
GET,CLAM,ID=TWXDATA.
SAVE,CLAM=CLAW1,ID=BATCHIN.
RETURN,CLAM.
GET,CLKM,ID=TWXDATA.
SAVE,CLKM=CLKW,ID=MASTER.
RETURN,CLKM.
GET,R2MM,ID=TWXDATA.
SAVE,R2MM=R2MW1,ID=MASTER.
SAVE,R2MM=R2MW1,ID=DATA12.
SAVE,R2MM=R2MW1,ID=DATA22.
RETURN,R2MM.
GET,R4MM,ID=TWXDATA.
SAVE,R4MM=R4MW1,ID=DATA12.
SAVE,R4MM=R4MW1,ID=DATA22.
RETURN,R4MM.
GET,RABM,ID=TWXDATA.
SAVE,RABM=RABW1,ID=DATA11.
SAVE,RABM=RABW1,ID=DATA21.
RETURN,RABM.
GET,RAPM,ID=TWXDATA.
SAVE,RAPM=RAPW1,ID=DATA11.
```

AD-A115 697 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 15/7
THEATER WARFARE PROGRAMS AT AFIT: AN INSTRUCTIONAL AID.(U)
MAR 82 A WAISANEN
UNCLASSIFIED AFIT/GST/OS/82M-15 NL

2 OF 2

END
DATE
FILMED
7-82
DTIC

```
SAVE,RAPM=RAPW1,ID=DATA21.
RETURN,RAPM.
GET,RBLM,ID=TWXDATA.
SAVE,RBLM=RBLW1,ID=DATA12.
RETURN,RBLM.
GET,RCRM,ID=TWXDATA.
SAVE,RCRM=RCRW1,ID=BATCHIN.
RETURN,RCRM.
GET,RLGM,ID=TWXDATA.
SAVE,RLGM=RLGW1,ID=BATCHIN.
RETURN,RLGM.
GET,RLUM,ID=TWXDATA.
SAVE,RLUM=RLUW1,ID=BATCHIN.
RETURN,RLUM.
GET,RACM,ID=TWXDATA.
SAVE,RACM=RACW,ID=MASTER.
RETURN,RACM.
GET,RMRM,ID=TWXDATA.
SAVE,RMRM=RMRR1,ID=DATA11.
SAVE,RMRM=RMRR1,ID=DATA21.
RETURN,RMRM.
GET,RMUM,ID=TWXDATA.
SAVE,RMUM=RMUW,ID=MASTER.
RETURN,RMUM.
GET,RPPM,ID=TWXDATA.
SAVE,RPPM=RPPW1,ID=BATCHIN.
RETURN,RPPM.
GET,RRLM,ID=TWXDATA.
SAVE,RRLM=RRLW1,ID=DATA22.
RETURN,RRLM.
GET,RSTM,ID=TWXDATA.
SAVE,RSTM=RSTW1,ID=BATCHIN.
RETURN,RSTM.
GET,RTGM,ID=TWXDATA.
SAVE,RTGM=RTGW1,ID=BATCHIN.
RETURN,RTGM.
GET,RWXM,ID=TWXDATA.
SAVE,RWXM=RWXW,ID=MASTER.
RETURN,RWXM.
REVERT.
```

```
.DATA,GO.
.PROC,GO,N.
BUILD,ID=DATA1#_N.
BUILD,ID=DATA2#_N.
REVERT.
.EOR
.DATA,JOB.
TWX,T500,I01000,CM100000.  T800855 FOX
BEGIN,NOSFIL.
GET,TWXGO,ID=TWXRUN.
GET,TWXLIB,ID=TWXPROG.
TWXGO.
REQUEST,LGO,*PF.
GET,AR,ID=TWXPROG.
FTN5,I=AR,LO=0.
CATALOG,LGO,ARLGO,ID=T800855,XR=FOX,PW=FOX.
RETURN,LGO,AR.
GET,LB,ID=TWXPROG.
REQUEST,LGO,*PF.
FTN5,I=LG,LO=0.
CATALOG,LGO,LBLGO,ID=T800855,XR=FOX,PW=FOX.
RETURN,LGO.
GET,MISEG,ID=TWXRUN.
GET,MI,ID=TWXPROG.
MISEG.
GET,APESEG,ID=TWXRUN.
GET,APE,ID=TWXPROG.
APESEG.
GET,LISEG,ID=TWXRUN.
GET,LI,ID=TWXPROG.
LISEG.
GET,INPTGO,ID=TWXRUN.
INPTGO.
GET,PROCGO,ID=TWXRUN.
PROCGO.
.EOF
```

Appendix E

TWX File Descriptions

Appendix E

TWX File Descriptions

Four indirect library files are used for storing all the data files, procedure files, programs and utility routines required by the TWX at AFIT. These library files are TWXDATA, TWXRUN, TWXPROG and TWXUTIL, respectively. The primary reason behind separating the files was to emphasize the difference in the files as much as possible. This separation method is equivalent to the file-string identifiers used for the TWX files at Maxwell AFB.

TWXDATA

All data files are stored in the indirect library file TWXDATA (Table VI). There are two types of data files, sequential card image and direct access binary. The following nomenclature has been adopted to distinguish between them

ABBx#

where:

A = C if the file is sequential (card image)
    R if the file is random (direct access)

BB = file identifier

x = M if the file is a master file
    W if the file is a working file

# = seminar number

TWXDATA contains only file masters. During initialization, copies are made of these files and are placed in the IFS files MASTER, DATA11, DATA12, DATA21, DATA22, and BATCHIN. MASTER contains those data files which are not modified during the execution phase. All IFS files with the prefix "DATA" contain the modifiable input files to APE, MI and LI where the name is of the form

DATAxy

where:

x = side (1 for Blue and 2 for Red)

y = 1, 2 or 3

1 = input files to APE and LI
2 = input files to MI (output from APE)
3 = output files from MI and LI

BATCHIN contains the input files to the batch jobs and the modified data files. Prior to the next day's run, those files which were originally in the DATA files (RAPW#, RABW#, R2MW#, R4MW#, RBLW# and RRLW#) must be modified with the appropriate utility programs and replaced.

TABLE VI

TWXDATA Files

Data File	Description
CAKM	Air constants file
CCAM	Combined actions file
CLAM	Land actions file
CLKM	Land constants file
RABM	Airbase file
RACM	Aircraft file
RAPM	AAFCE planning file
RBLM	Blue land file
RRLM	Red land file
RCRM	Combined reconnaissance file (not required for theater simulation)
RLGM	Logistics file
RLUM	Land units file
RMUM	Munition load file
RMRM	Mission reference file
R2MM	Mission working master file (2ATAF)
R4MM	Mission working master file (4ATAF)
RMWM	Mission working master file (ATAF flag not set)
RPPM	Preprogrammed control file (initially blank filled)
RSTM	Statistics file (initially blank filled)
RTGM	Target file
RWXM	Weather file

TWXPROG and TWXRUN

Two files are used for storing programs; TWXPROG and TWXRUN. TWXPROG contains only those programs (Table VII) which are totally source code (AR, APE, MI, LI, LB and TWXLIB). Programs in TWXRUN (Table IIIX) are preceded by a procedure file which compiles the program, accesses the necessary data files, executes the program, and replaces the modified data files. These programs are the relatively small batch programs SQ, MR, AG, MA, OR and LA. The purpose of this separation is to maximize clarity and minimize storage space. The primary function of TWXRUN is the storage of procedure files. Since the relatively small batch programs are prefixed with procedure files, they are stored in TWXRUN. The object decks for each of the small batch programs could be created once and stored on a disk file but then storage space would be required for two procedure files, the program source and its object deck for each of the programs. This way, only one procedure file and source is stored for each program.

TABLE VII

TWXPROG Files

File	Description
APE	AAFCE Planning Executive program
AR	Air battle simulation program
LB	Land battle simulation program
LI	Land order input program
MI	Mission order input program
MISC	Unmodified utility programs
TWXLIB	TWX Library routines
XXSRCE	Original main driver routine XX

TABLE IIX

TWXRUN Files

File	Description
AGGO, AGGOB	AG execution proc
APEGO	APE execution proc
APESEG	APE segmentation proc
ARGO	AR batch job spawning proc
ARSEG	AR segmentation proc (unvarified)
ARSEGO	AR segmentation spawning proc
INPTGO	INPTR compilation and cataloging proc
IPGO	IP execution proc
LAGO, LAGOB	LA execution proc
LBGO	LB execution spawning proc
LBSEG	LB segmentation spawning proc
LBSEG	LB segmentation proc (unvarified)
LBSEGO	LB segmentation spawning proc
LGGO	LG execution proc (no longer used)
LIGO	LI execution proc
LISEG	LI segmentation proc
MAGO, MAGOB	MA execution proc
MIGO	MI execution proc
MISEG	MI segmentation proc
MR	MR execution proc
ORGO, ORGOB	OR execution proc
SETUP	TWX file initialization proc
SQGO	SQ execution proc
STGO	ST execution proc
TWXJCL	original Honeywell JCL (archive file)
WSSGO	WSS execution proc

TWXUTIL

During the execution of the TWX, various data files must be modified, some directly, others indirectly. Unlike the sequential card image files which can be printed, edited, and replaced, direct access files require programs for reading and editing. Of the 22 data files used in TWX, 18 are direct access (these are identified by the first character; "R" for random). For these data files, programs have been developed to permit editing (Table IX).

TABLE IX

TWXUTIL Files

Program	File(s) read and Modified
RABREAD	RABx#
RACREAD	RACx
RAPREAD	RAPx#, RPPx# (no changes permitted)
RBLREAD	RBLx#, RRLx#, RMUx
RCRREAD	RCRx#
RLUREAD	RLUx#
RMRREAD	RMRx# (no changes permitted)
RPPREAD	RPPx#
RTGREAD	RTGx#
R2MREAD	R2Mx#, R4Mx#, RLGx and RMRx# (logicals are displayed as integers)

Vita

Anthony Waisanen was born in Warren, Minnesota on 7 January 1953 to Einard and Doris Waisanen. After graduating from Wadena Senior High School, Wadena, Minnesota in 1971, he enrolled at Concordia College, Moorhead, Minnesota. A year later, he enrolled at the University of Minnesota, Duluth where he received a commission in the Air Force through ROTC and Bachelor of Science degrees in Mathematics and Biology in June 1976. Following graduation, he was employed as a conventional weapon systems analyst at Eglin AFB, Florida from 24 October 1976 to 13 July 1980 before entering AFIT in September 1980. He is married to the former Holly A. Eifert of Wadena, Minnesota.

Permanent address: Box 116
Verndale, Minnesota 56481

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GST/OS/82M-15	2. GOVT ACCESSION NO. AD-A115697	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THEATER WARFARE PROGRAMS AT AFIT: AN INSTRUCTIONAL AID		5. TYPE OF REPORT & PERIOD COVERED M.S. Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Anthony Waisanen Capt USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB OH 45424		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB OH 45424		12. REPORT DATE March 1982 13. NUMBER OF PAGES 108
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

4 JUN 1982

Lynn Wolaver
LYNN E. WOLAVER
Dean for Research and
Professional Development
AIR FORCE INSTITUTE OF TECHNOLOGY (ATC)
WRIGHT-PATTERSON AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computerized simulation	Operations research
Programming techniques	Central Europe scenario
Gaming models	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis report details the processes and modifications that were required in order to execute the Theater Warfare Exercise (TWX) on the CDC computers at Wright-Patterson AFB. Prior to this effort, the TWX programs and data files could only be accessed and executed with Honeywell computers.

By modifying the data files, program coding, overlaying techniques, and operation, the TWX can now be run on any CDC computer. Any other computer with sufficient central memory and an ANSI standard FORTRAN-77

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

BLOCK 20.

compiler can also execute the programs provided the operation methods (procedure files) are modified).